THÈSE

ARCHITECTURAL EXPLORATION METHODS AND TOOLS FOR HETEROGENEOUS 3D-IC

Présentée devant
L’Ecole Centrale Lyon

Pour obtenir le grade de
Docteur de l’Ecole Centrale

Formation doctorale : Conception de dispositifs de l’électronique intégrée
École doctorale : Électronique, Électrotechnique et Automatique de Lyon
Thèse préparée à l’Institut des Nanotechnologies de Lyon, UMR CNRS 5270

Présentée et soutenue par :
Felipe Frantz Ferreira

Soutenue le 26 Octobre 2012 devant la Commission d’examen

Jury MM.

Frédéric Pétrot ................................................. Président
Professeur des Universités (TIMA – INP Grenoble)
François Pécheux ........................................ Rapporteur
Maître de Conférences, HDR (LIP6 – UPMC)
Helmut Gräb .................................................. Rapporteur
Privatdozent, HDR (TU-München)
Ian O’Connor ................................................... Directeur de thèse
Professeur des Universités (INL – Centrale Lyon)
Marc Belleville ............................................ Examinateur
Docteur, HDR (CEA – LETI)
Yves Leduc ..................................................... Examinateur
Professeur Chaire Texas Instruments (Polytec’Nice-Sophia)
Dominique Marron ........................................ Membre Invité
Ingénieure (ST-Microelectronics)
Lioua Labrak .................................................. Membre Invité
Chargé de Recherche (INL – Centrale Lyon)
CONTENTS

1. General introduction 1
   1.1. A new hardware paradigm: Heterogeneous 3D integration . . 1
   1.2. Designing for 3D integration: problems and motivations for the thesis ......................... 6
   1.3. Context and structure of the thesis ....................... 8

PART I Tools for 3D-IC partitioning

2. Introduction 13
   2.1. State of the art .............................. 13
   2.2. Discussion and motivation .......................... 16

3. MOFP - An extensible partitioning tool 18
   3.1. Tool architecture ............................... 18
   3.2. A meta-optimization approach for parameter tuning ........... 24

4. Modularity 29
   4.1. Area-Wirelength problem in 3D .......................... 29
   4.2. Thermal-aware partitioning ............................. 34
   4.3. MoNICA - Traffic optimization for 3D NoC-based MPSoCs . 40
   4.4. Summary ..................................... 47

5. Experimental Results 48
   5.1. Implementation and test environment ........................ 48
   5.2. Automatic parameter tuning with GA ....................... 48
   5.3. Thermal-aware floorplanning .............................. 55
   5.4. Network-aware floorplanning .............................. 59

6. Conclusion 63
   6.1. Summary ...................................... 63
   6.2. Perspectives .................................. 64
PART II Hierarchical design of heterogeneous systems

7. Introduction 69
   7.1. Basic concepts in automated design .................... 70
   7.2. A review of hierarchical design methods ................. 77
   7.3. Discussion and motivation ................................ 81

8. Predictive hierarchical synthesis using Pareto-fronts 85
   8.1. Design flow .............................................. 85
   8.2. Pareto-front computation by numerical optimization .... 95
   8.3. Building predictive models .............................. 103
   8.4. Accounting for environment parameters .................. 119
   8.5. Summary .................................................. 121

9. Experimental results 122
   9.1. Implementation and experimental method .................. 122
   9.2. Data link for optical NoC application .................... 124
   9.3. Operational Transconductance Amplifier (OTA) .......... 130
   9.4. Non-monotonic Pareto Front ............................. 133

10. Conclusion 138
    10.1. Summary ................................................ 138
    10.2. Perspectives ............................................ 139

11. General conclusion 141

List of publications 143

Bibliography 144

List of figures 161

List of tables 164

Acronyms 165
ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my research adviser Professor Ian O’Connor and to Dr. Lioua Labrak. Their experience, generosity and energy were instrumental to the successful accomplishment of this work.

I would like to thank Professor Frédéric Pétrot, of the Grenoble-INP (Institut Nationale Polytechnique), for accepting the role of president of the jury.

My gratitude also goes to Professor Helmut Gräb of the Technische Universität München and to Professor François Pécheux of the Université Pierre et Marie Curie, for agreeing to be my referees, and who have devoted much time and energy to the assessment of this thesis.

I address my thanks to Professor Yves Leduc, of Polytec’Nice-Sophia, to Dr. Marc Bellevile, of CEA-Leti and Dr. Dominique Marron, of ST Microelectronics, for consenting to participate in this jury.

These three years at INL were years of intense professional and personal development. I express my gratitude to all those who participated in this development and specially to Wan Du and Vijayaragavan Viswanathan for their generosity and friendship.

I dedicate this work to my family, who always supported me in every choice I made.
3D integration technology is driving a strong paradigm shift in the design of electronic systems. The ability to tightly integrate functions from different technology nodes (analog, digital, memory) and physical domains (MEMS, optics, etc) offers great opportunities for innovation (More than Moore). However, leveraging this potential requires efficient CAD tools to compare architectural choices at early design stages and to co-optimize multiphysics systems.

This thesis work is divided into two parts. The first part is dedicated to the problem of partitioning a system into multiple dies. A 3D floorplanning tool was developed to optimize area, temperature and the interconnect structure of a 3DIC. Moreover, a meta-optimization approach based on genetic algorithms is proposed to automatically configure the key parameters of the floorplanner. Tests were carried out on architectural benchmarks and a NoC based multiprocessor to demonstrate the efficiency of the proposed techniques.

In the second part of the thesis, a hierarchical design methodology adapted to heterogeneous systems is presented. The method combines the bottom-up and top-down approaches with Pareto-front techniques and response surface modeling. The Pareto front of lower level blocks are extracted and converted into predictive performance models that can be stored and reused in a top-down optimization process. The design flow is demonstrated on an operational amplifier as well as on the synthesis of an optoelectronic data link with three abstraction levels.
RéSUMÉ


La première partie de la thèse est dédiée au problème de partitionner la fonctionnalité d’un système sur de multiples puces. Un outil de “floorplan” 3D a été développé pour optimiser ce partitionnement en fonction de la surface des puces, de la température d’opération du circuit et de la structure des interconnexions. Ce type d’outil étant complexe, nous proposons de régler ses paramètres de façon automatique par l’utilisation d’algorithmes évolu- tionnaires. Des résultats expérimentaux sur une suite de benchmarks et sur une architecture multi-processeur connecté en réseau démontrent l’efficacité et l’applicabilité des techniques d’optimisation proposées.

Dans la deuxième partie, nous présentons une méthodologie de conception hiérarchique qui est adaptée aux systèmes hétérogènes. La méthode combine une approche ascendante et descendante et utilise des courbes de compromis (Fronts de Pareto) comme une abstraction de la performance d’un circuit. La contribution principale de la thèse consiste à utiliser des techniques d’interpolation pour représenter les Fronts de Pareto par des fonctions continues et à leur intégration dans des processus d’optimisation classiques. Cela permet un gain en flexibilité lors de l’étape ascendante du flot (caractérisation) et un gain en temps lors de l’étape descendante (synthèse). Le flot de conception est démontré sur un amplificateur opérationnel ainsi comme sur la synthèse d’un lien optoélectronique avec trois niveaux hiérarchiques.
1 General introduction

1.1 A new hardware paradigm: Heterogeneous 3D integration

Moore’s law scaling, the reduction of minimum device feature size to improve speed and integration density in planar integrated circuits, is increasingly difficult to continue due to major economic and physical obstacles. Three-dimensional (3D) integration is an emerging technology that can form highly integrated systems by vertically stacking and connecting various materials, technologies and functional components together. By exploiting the vertical dimension, it provides an opportunity to continue to achieve the performance levels predicted by the extrapolation of Moore’s law, but using a different technological approach. While by no means technologically simple, it does represent a major design paradigm shift away from the conventional System-on-Chip (SoC) approach of traditional Moore’s law scaling towards equivalent scaling and functional diversity through unconventional System-in-Package (SiP) approaches (Figure 1.1).

Research has considered the integration of different silicon dies for many years [BOTW72]. However, it has not been until the last decade that die stacking has started to become a reality in products (e.g. high-performance server products [PGS+98]). It is now widely recognized that the industry paradigm will shift to a new industry-fusing technology era that will offer tremendous global opportunities for expanded use of 3D silicon-based technologies in highly integrated systems. According to the CEO of Samsung, "We are at the doorstep of the largest shift in the semiconductor industry ever, one that will dwarf the PC and even the consumer electronics era ... The core element needed to usher in the new age will be a complex integration of different types of devices such as memory, logic, sensor, processor and software, together with new materials, and advanced die stack technologies, ... Rapid adoption of 3D integration technology seems to be essential and, thankfully, unavoidable."\footnote{Dr. Chang-Gyu Hwang, president and CEO of Samsung Electronics, speaking at the 2006 IEEE International Electron Devices Meeting (IEDM)}
1.1.1 Principle of 3D integration: approaches and limitations

Advances in packaging trends enable various interconnect solutions for SiP. One possible approach is the use of perimeter wire bonding (WB) [KH07]. Elpida\(^2\), a memory company, currently uses WB to connect up to four layers of DRAM chips in a super-thin package (0.8mm) targeting the smart phone industry. The benefits of WB are mainly usage of a known technology (Figure 1.2). The disadvantages are parasitic effects (especially inductances) and significant surrounding area for a stack of chips. In addition, as WB is restricted to peripheral connections, it is not suitable for systems with a very large number of interconnect lines.

An emerging technology for 3D integration is Through Silicon Vias (TSV) [KPA+06]. This technique permits direct silicon-on-silicon stacking (Figure

\(^2\)http://www.a-elpida.com

Figure 1.1: Miniaturization vs. diversification (adapted from [ITR09])

Figure 1.2: 3D Packaging technologies (adapted from [MEN])
1.1. A NEW HARDWARE PARADIGM: HETEROGENEOUS 3D INTEGRATION

Figure 1.3: Example of a vertical stacking with 3 ICs (a) and the cross-cut of a medium density TSV (b), source [3DI10].

1.3). The metallic via traversing the silicon connects the regular interconnect layers (back end of line – BEOL) to the backside of the chip. A redistribution layer (RDL) then routes the signal to another connection type (e.g. solder bump) suitable for bonding. TSVs can be formed before or after the BEOL (Figure 1.4). In both cases, a subsequent thinning of the device-wafer bares the metallized TSVs.

TSVs present clear advantages over WB: (i) they are not restricted to peripheral connections, thus increasing the I/O density; (ii) they allow a dramatic reduction in wirelength and (iii) are less subject to parasitics [PF09].

However, the technology is not yet as mature as WB and further research has to be carried out [DAC+10]. Leading processor and memory companies such

"Via-First" approach

"Via-Last" approach

Figure 1.4: Formation of TSVs (adapted from [Yol07])
as IBM [TJS+06] and Intel\(^3\) are actively pursuing such programs. Tezzaron\(^4\), another USA based technology company, claims to have plans for fabrication of prototypes with stacks of multiple 13\(\mu\)m thick wafers. Technological issues remain, such as TSV density (< 5\(\mu\)m pitch), thermal management, mechanical stability of thin (between 10-15\(\mu\)m) die handling and bonding. It is however very clear that TSV will be the main technology enabling the 3D integrated circuit (3D-IC) revolution [Ark11].

1.1.2 Key benefits and driver applications

The possibility to stack dies and replace off-chip wires by TSVs has a series of benefits, which include [DWM+05]:

- **Improved performance metrics**: balance of the communication latencies on a single die and within the complete chip. The bandwidth bottleneck imposed by limited pin count and long interconnects on a board or a substrate in a 2D implementation can be eliminated. For example, the WideIO standard for memories stacked on top of processors can reach transfer rates up to 17GBps, i.e. two times more than the LPDDR2 standard [WID].
- **Enhanced miniaturization**: the real estate available for circuits and logic is optimally used by building upwards, while the overall footprint of the packaged system is reduced. For instance introducing a quadratic shape of the used dies minimizes the length of the die interconnect.
- **Reductions in energy and power**: this results from the lower average energy per signal switching transfer due to shorter wires. The silicon area and power consumed by repeaters can be reduced [DOSF09].
- **Enabling of heterogeneous technology integration**: Analog, RF components and I/O circuitries scale badly at advanced technology nodes and their size can even increase to compensate for inadequate technological characteristics for their needs. Separating these functions from logic allows to reduce costs and boost performance. More importantly, die stacking provides an easy way to integrate incompatible technologies such as MEMS, photonics, high-voltage electronics and bio-sensors.

\(^3\)http://www.intel.com/research/platform/terascale/teraflops.htm
\(^4\)http://www.tezzaron.com
1.1. A NEW HARDWARE PARADIGM: HETEROGENEOUS 3D INTEGRATION

Figure 1.5: Roadmap for TSV technology (adapted from [O’C09])

- Higher yield: a multitude of small dies has a higher yield than a single large die. Moreover, heterogeneous integration allows to simplify and optimize the processes on each wafer (e.g. logic and memory).

Some technical disadvantages also exist: thermal evacuation of the heat generated in the inner layers is poor. Secondly, the miniaturized connections make test difficult [Mar10].

The 3D technology and its associated supply chain are being developed around the so-called “killer applications”, those that will benefit most of 3D. Figure 1.5 illustrates the sequence of applications that will drive the miniaturization of the vertical interconnections. Smart cameras, with CMOS image sensors on top of processors are already in production (e.g. the VD6725 chip from ST Microelectronics). Stacks with identical memory chips will come next to be later stacked on processors. It is expected that the increased spatial proximity between the processor and the memory will enable new addressing and information search schemes (e.g. by higher parallelism). The ultimate application of 3D-IC are highly integrated smart sensors. Another on-going revolution, that of the Internet-of-Things [Eva11], is pushing the market of sensors to a spectacular growth in the coming years (Figure 1.6). Stacks combining memory, processor, sensors, analog, RF, and power harvesting modules will create all sorts of intelligent and communicating devices that interact with the user and the surrounding environment.

5http://www.st.com
1.2 Designing for 3D integration: problems and motivations for the thesis

Although 3D integration shows promise, significant challenges associated with efficient circuit and system design hamper its adoption and further development. Without tools to allow exploration of design trade-offs to be made in 3D-ICs, engineers are restricted to design in two dimensions and occasionally stack chips crudely. Conversely, without one clear market for 3D design, EDA vendors are unlikely to offer tools. A critical challenge from the design point of view is therefore to be able to extend existing design and simulation approaches to take into account the 3D integration paradigm. Important aspects relative to 3D-IC simulation and design are pointed out next.

The wafer thinning process and the different coefficients of expansion between silicon and the metal filling the TSVs induce mechanical stress. This stress around a TSV modifies the electrical properties of the substrate in short range and is another source of variability in the devices [SKC+11].

From the perspective of signal integrity, the electromagnetic properties of the TSV have to be characterized according to the kind of via (e.g. filled or hollow) and silicon substrate (e.g. low or high resistivity). Related works comprise experimental measurements and the derivation of compact and analytical models for both the TSVs and the interconnections running through RDLs [KPC+11, AEDV+11, FLC+11].
Finally, heat dissipation is considered the major issue in 3D [SVR+10], since the tight packaging increases the power density and limits the surface in contact with the heat sink [WFT04]. The direct relationship between temperature, leakage current and reliability, makes it one of the most studied effects in the 3D-IC design [SSH+03, KZBS06, KKHH11, MTH+12].

These physical effects affect all steps of the design process. We summarize the needs for adequate tools for both phases of system design:

1. Physical design: CAD tools are required to allow flexible floorplanning, with vertical and horizontal place and route steps, including thermal and reliability constraints. There is also a need to be able to handle the sheer amount of data that goes into the physical design (multiple layers, multiple technologies).

2. Architectural design: strategic technology decisions must be made early in the architecture planning process rather than as a packaging decision after circuit design is complete. This requires taking the 3D design space into account right from the start of system design in order to distribute its different parts into a new set of chips that will be stacked [CJM+06, LMK+07, WPZT09].

1.2.1 Motivations

The opportunity to improve a product by splitting its functionality over several dies puts system-level partitioning at the heart of the 3D-IC design flow. System architects need to investigate which combination of technologies and physical partitioning of the system will lead to the best gains, which can be a reduction of the IC form factor or manufacturing cost, an improvement in performance or a combination of all these. The architect should also look towards solutions that mitigate the disadvantages imputed to 3D integration, such as complex test and poor heat evacuation. This complex trade-off analysis has to be carried out in the very early design steps, by combining adequate tools and experience from previous designs.

In this thesis, we focus on this field to examine how to deal with some of the complexity of 3D-IC design. Since 3D integration is based on the spatial organization of functions across dies, one of our research interests is on partitioning, and more accurately, 3D floorplan optimization. Another
CHAPTER 1. GENERAL INTRODUCTION

The topic of this research is design methodologies for heterogeneous systems. With 3D-IC facilitating the diversification of functionalities, it is pivotal to bring analog/mixed-signal (AMS) design to the same productivity levels as digital design.

1.3 Context and structure of the thesis

1.3.1 Context: CATRENE 3DIM³ project

The 3DIM³ project (3D-TSV Integration for Multimedia and Mobile Applications) is a CATRENE approved proposal bringing together nine companies (semiconductor manufacturers, design houses and EDA vendors) and six academic and research institutions from France, Netherlands and Germany. It aims at providing novel system design methodologies, new design tools and system architecture solutions to handle this emerging 3D TSV integration technology and proposes to enable the efficient design, from system level to layout, of 3D integrated multimedia and mobile products with higher performances, lower power, and smaller size/form factor at lower cost. This thesis was funded over a period of three years by this project, and contributed to two tasks: T1.2 Definition of model abstraction for 3D integration – hierarchical modeling concept, and T4.5 Methods and tools for system-level design.

1.3.2 Structure and contributions

The thesis covers two main topics: chip partitioning and system design methodology. These topics are presented separately, dividing the document into two parts. Each part is structured in a sequence of chapters. They cover a deeper introduction to the topic, the contributions of this work, experimental results, conclusions and perspectives. The main contributions of the thesis are summarized as follows:

Part I – Tools for 3D-IC partitioning – The core of the work is the development of a flexible 3D floorplan optimization tool based on the Simulated

http://www.catrene.org – Cluster for Application and Technology Research in Europe on NanoElectronics
Annealing (SA) meta-heuristic. This tool will be incrementally extended to incorporate the thermal dimension and, later, to co-optimize the traffic in a Network-on-Chip (NoC). To achieve high floorplan quality and simplify the implementation of the tool, a novel meta-optimization approach based on Evolutionary Algorithms (EA) is proposed to fine-tune the optimization parameters of the floorplan algorithm automatically. This frees the software engineer from a manual and empirical task. The structure of the floorplan tool and the meta-optimization process are presented in chapter 3. The progressive extensions of the tool to deal with more complex floorplan problems are presented in chapter 4: starting with the basic area-wirelength problem we add temperature effects and, in collaboration with researchers from CEA-LETI and UFRGS, we develop a tool dedicated to the partitioning of 3D chips interconnected by NoCs. For each of these problems, we propose new heuristics and strategies to improve the optimization process. These new strategies are validated in chapter 5 using benchmark circuits.

**Part II – Hierarchical design of heterogeneous systems** – We propose a design framework enabling the exploration of trade-offs at system-level and their automatic hierarchical synthesis. The work is based on the emerging technique of using performance trade-off curves (Pareto Fronts) as an abstraction for the behavior of a component. This technique is marked by the characterization of trade-offs of low-level blocks and their bottom-up propagation to compute the trade-offs of a system. Motivated by the opportunity to leverage the data collected during the characterization process, we propose to interpolate this design data and create a continuous mapping of the trade-offs across the hierarchy, called a predictive model. By interpolating the data in both the performance and the parameters spaces, we are able to go from system-level performance requirements down to a prediction of the sizing of low-level blocks. This framework is described in chapter 8, where we formalize the predictive synthesis flow and show how the predictive models integrate seamlessly into typical optimization-based design processes. Additionally, we detail the steps and mathematical tools needed to make the predictive model be a self-contained and language-agnostic representation of functional block. Three experiments are discussed in chapter 9, where we investigate the quality of the predicted designs. Two experiments relate to real-world circuits: the hierarchical design of an optical data link and an operational transconductance amplifier.
A general conclusion is provided in chapter 11 where we expose some perspectives to bring the tools developed in this work together and create a holistic design flow for heterogeneous 3D-IC.
2. Introduction

2.1. State of the art

2.1.1. Floorplan optimization methods

2.1.2. 3D-IC partitioning

2.2. Discussion and motivation

3. MOFP - An extensible partitioning tool

3.1. Tool architecture

3.1.1. Elements subject to parameterization

3.2. A meta-optimization approach for parameter tuning

3.2.1. Pareto optimality and multiobjective GA

3.2.2. Problem formulation

3.2.3. Incremental optimization and validation

4. Modularity

4.1. Area-Wirelength problem in 3D

4.1.1. Area evaluation

4.1.2. Accounting for TSV dimensions

4.1.3. Problem formulation

4.1.4. Moves

4.2. Thermal-aware partitioning

4.2.1. Temperature evaluation

4.2.2. Problem formulation - A two-phase optimization

4.2.3. Moves - The power density heuristic
4.3. MoNICA - Traffic optimization for 3D NoC-based MPSoCs  
   4.3.1. Problem formulation - A nested optimization process  
   4.3.2. NoC-aware floorplan  
   4.3.3. Floorplan refinement  
4.4. Summary  

5. Experimental Results  
   5.1. Implementation and test environment  
   5.2. Automatic parameter tuning with GA  
      5.2.1. Configuring the meta-optimization problem  
      5.2.2. Validation with benchmarks  
      5.2.3. Accounting for TSV dimensions  
   5.3. Thermal-aware floorplanning  
   5.4. Network-aware floorplanning  

6. Conclusion  
   6.1. Summary  
   6.2. Perspectives
2 INTRODUCTION

This part covers some of the various aspects of 3D-IC partitioning. The core of the work is the development of a flexible 3D floorplan optimization tool. This tool will be extended to incorporate the thermal dimension and, later, specialized to co-optimize the traffic in a Network-on-chip (NoC). To achieve high floorplan quality and simplify the implementation of the tool, a novel meta-optimization approach based on Evolutionary Algorithms (EA) is proposed to fine-tune the optimization parameters of the floorplan algorithm automatically, freeing the software engineer from a manual and empirical task.

The contents in this part are organized as follows. The next section reviews the important contributions partitioning and optimization of 3D chips. It starts with the basics of 2D floorplan and then reviews the works related to 3D integration. Then the contributions of this work are discussed in separate chapters. Chapter 3 introduces the basic structure of the floorplan tool. Based on this structure, the meta-optimization approach is presented in section 3.2. Chapter 4 is dedicated to the extensions supporting temperature minimization and traffic optimization in NoCs. The experimental results are presented and analyzed in chapter 5.

2.1 State of the art

2.1.1 Floorplan optimization methods

The fundamental floorplan problem is to lay out a collection of blocks (shapes of different sizes) so that no two blocks overlap and that the occupied area and the length of the interconnections are minimal. While the problem is conceptually easy, its automatic solution via numerical optimization requires an efficient manner to guarantee the no-overlap condition. This is commonly achieved by restricting the admissible position of the blocks. They cannot be freely positioned on a plane, but need to be aligned with the edges of
other blocks (Figure 2.1). Positions can then be expressed relatively as topological relations (e.g., right to, above of) and captured by vertical and horizontal constraint graphs. Translating the topological representation back to Cartesian coordinates is performed using a so-called packing algorithm that guarantees no-overlap by traversing the graphs in appropriate order. By representing the placement problem in such way, finding the optimum falls into a combinatorial problem, where the relations between blocks are iteratively permuted.

Solving this combinatorial problem dates back from the early 80’s with the seminal works of Kirkpatrick et al. [KGV83], introducing the Simulated Annealing (SA) heuristic and Wong and Liu [WL86] presenting the Normalized Polish Expression (NPE) to describe slicing floorplans. Since then, many works were published proposing other topological representations and their corresponding packing algorithms. With each new representation, researchers tried to reduce redundancies in the design space, create faster packing algorithms or add mechanisms to describe placement constraints directly in the constraint graph.

Popular representations include the Sequence-Pair (SP) [MFNK96], the B*-tree [CCWW00] and the Transitive Closure Graph (TCG) [LC05]. Sequence-Pair encodes topological relations using two permutation lists and is, by far, the easiest to manipulate. B*-tree is based on binary trees and always leads to compact solutions, although its reduced design space might exclude optimal solutions for wirelength. The TCG is equivalent to SP, but differs on where the computational effort is concentrated. In SP, each permutation of elements in the lists is performed in \( O(1) \) time, but a new constraint graph is built from scratch to compute \( xy \) coordinates. The TCG, on the other
2.1. STATE OF THE ART

hand, is directly based on the constraint graph, therefore blocks positions are directly available but making incremental changes to the graph must follow complex rules to keep consistency.

With the maturity of the field, the attention has turned away from the underlying data structure and towards the optimization procedure [CAM05]. Random swap and rotation of blocks have been replaced by more intelligent strategies that can converge faster towards the desired result and give predictable results.

A major contribution to this topic is the work at the University of Michigan, USA, on hierarchical floorplanning and improved local search with its open source tools Parquet (floorplan) and CAP0 (placement) [AM03, RAPM06]. We can also cite the work of Shiu and Lim [SREL04] on penalty functions to satisfy reliability constraints and the recent works on the use of hierarchical binary trees to express symmetry and regularity in analog circuits [SEG+08, COC11].

The majority of the floorplanners are based on the Simulated Annealing heuristic, but other approaches exist. For example, tools like 3D-STAF (floorplan) [ZML+07] and Kraftwerk-2 (placement) [SSJ08] use a force-directed approach. These methods solve a continuous optimization problem, where the blocks move freely in space and are attracted (or repelled) by the others until a state of equilibrium is found. Block overlap is allowed in the first iterations and is progressively solved by an increasing penalty factor. Apart from this, in a work dedicated to analog placement with constraints, Strasser et al. [SEG+08] present Plantage a deterministic layout method that can efficiently enumerate feasible solutions with different aspect ratios using a bottom-up hierarchical approach.

All these latter publications witness a diversification in the floorplanning problem to answer specific questions. And, although adapting the underlying data structures to represent 3D stacks is not a problem, co-optimizing the many issues that arise in 3D chip design and exploring the multiple integration scenarios is a challenge.

1 Chip floorplanning and analog placement are similar problems: both deal with a relatively low number of instances with different shapes and size. Conversely, digital placement deals with a large number of instances (gates) of similar sizes to be placed in rows.
2.1.2 3D-IC partitioning

The many works dedicated to the layout of 3D chips have covered individual aspects of the design. A great number deals with the minimization of temperature in the stack, either by floorplanning [CWZ04, ZML07, XSXY10] or by routing and via placement [ZZS06, GS06]. Other authors investigated the co-optimization of floorplan and power/ground networks [FXCW10], or still the placement under reliability constraints [SREL04]. All these tools make extensions to the basic floorplan representations (SP, B*-Tree, etc).

Another important concern of 3D chip partitioning is the communication infrastructure. The 3D-IC technology targets high performance applications, with many processing elements communicating in parallel. To enable scalable performance, it is important to use structured interconnects that allow efficient resource sharing, such as NoCs [SPJ11]. Tools dealing with NoCs and floorplanning are far less studied than thermal-aware floorplanners. They typically proceed in a sequential approach, first mapping an application into layers with min-cut algorithms and then optimizing the floorplan of each layer individually [dPA10, SMBDM10].

2.2 Discussion and motivation

The partitioning of 3D systems is a joint optimization of objectives such as minimal area, temperature and wirelength. According to the problem in hand, more specialized metrics may be of importance such as mechanical stress or the organization of the communication infrastructure.

An overwhelming number of tools is based on the SA heuristic to solve the optimization problem. This heuristic (as with all optimization algorithms) needs to be tuned to the specific problem to allow a fast and efficient convergence. This tedious task is generally not discussed, and most attention is devoted to the other crucial aspect of the optimization, the cost function.

Taking these points into account, the contributions of this work are:

- The development of a modular floorplan tool, named mofp, based on SA. The modular structure allows to incrementally add support to...
new aspects of 3D design and specialize the optimization strategies accordingly.

- The introduction of a meta-optimization approach to fine-tune the optimization parameters of mofp. This meta-optimization process uses a genetic algorithm (GA) and a set of “training benchmarks” to search for a configuration of the tool performing well in problems with different complexity.

- We extend the tool to deal with three design concerns of growing complexity: area-wirelength, area-wirelength-temperature and finally area-temperature-latency for a NoC-based 3D system. For each of the problems we introduce new formulations of the optimization problem. In particular, for the NoC-based system, we propose a nested optimization loop to constrain the general floorplan problem to respect the regularity of a NoC topology.

The structure of mofp and the principle of meta-optimization are presented in chapter 3. The implementation of modules adding more aspects of 3D design are discussed in chapter 4. For each module, we highlight the novelties we introduce. Finally, in chapter 5 we validate the proposed methods through their application on benchmark circuits.
3 MOFP - AN EXTENSIBLE PARTITIONING TOOL

Considering the diverse aspects in 3D-IC design, a relevant partitioning tool must co-optimize a variety of objectives. This chapter presents mofp (multiobjective floorplanner), the 3D floorplanning tool implemented in this thesis (Figure 3.1). We discuss how its architecture is built around the SA heuristic and list which elements of the optimization framework are subject to be parameterized. To automate this parameterization we introduce a generic meta-optimization loop based on a GA.

3.1 Tool architecture

The mofp framework is based on Simulated Annealing (SA). SA is a stochastic optimization process in which up-hill moves (worsening of the solution) are accepted in an attempt to avoid premature convergence to local minima. It mimics the annealing process in metallurgy, where a material is cooled down slowly to allow its structure to settle in a state of minimum energy. By analogy, in SA, the energy of a solution is measured by a cost function and the probability of accepting an up-hill transition (typically a Boltzmann distribution) is controlled by the annealing temperature, which is progressively lowered according to a schedule.

![Diagram showing the workflow of the floorplan tool]

Figure 3.1: General workflow of the floorplan tool
Algorithm 1 Simulated Annealing algorithm

**Input:** Initial solution \( \{X, f\} \)

1: \( X_{\text{best}} \leftarrow X \), \( f_{\text{best}} \leftarrow f \)
2: \( T \leftarrow T_0 \)
3: while stop conditions not met do
4: \( X_{\text{next}} \leftarrow \text{modify}(X) \)
5: \( f_{\text{next}} \leftarrow \text{evaluate}(X_{\text{next}}) \)
6: \( \Delta_{\text{cost}} \leftarrow \text{compare}(f_{\text{next}}, f) \)
7: if \( \Delta_{\text{cost}} \leq 0 \) then # Always accept downhill moves
8: \( X \leftarrow X_{\text{next}}, F \leftarrow F_{\text{next}} \)
9: if \( \text{compare}(f, f_{\text{best}}) \leq 0 \) then
10: \( X_{\text{best}} \leftarrow X, f_{\text{best}} \leftarrow f \)
11: end if
12: else if \( e^{\Delta_{\text{cost}}/T} > \mathcal{U}(0,1) \) then # Accept some uphill moves
13: \( X \leftarrow X_{\text{next}}, f \leftarrow f_{\text{next}} \)
14: end if
15: \( T \leftarrow \text{update}(T) \)
16: end while
17: return \( X_{\text{best}}, F_{\text{best}} \)

The elements in this procedure are recalled in algorithm 1. In the context of this work, the decision variable \( X \) corresponds to the 3D floorplan using a topological representation and \( f \) is a vector of quality metrics (e.g. area, wirelength, maximum temperature). The reader can easily identify the classical steps of an optimization process: modify, evaluate and compare. To achieve the desired flexibility, mofp follows an object-oriented approach and provides API’s to extend each of these functions in order to include different aspects to the partitioning process. The overall architecture of the tool is described in Figure 3.2.

A frontend provides I/O capabilities, reading in design files into a database and creating a graphical view of the layout. The user inputs comprise the description of the system (essentially the dimensions and connectivity of its constituent blocks), the description of the stack (e.g. number of layers, stack order and TSV dimensions) and optimization directives (e.g. maximum number of iterations, cost function). Specific plugins can require additional inputs. For instance, the thermal plugin requires power information and the thermal properties of the stack.

The partitioning kernel encapsulates the implementation of the optimiza-
CHAPTER 3. MOFP - AN EXTENSIBLE PARTITIONING TOOL

The optimization engine delegates the modification and evaluation of the floorplan through requests to the move selection and the evaluation objects. These two objects offer entry points to extend the tool capabilities via plugins. The plugins register new simulation models in the evaluation object, which executes them in the given order and returns the quantified metrics to the annealer for cost computation. A plugin can also register strategies (called moves) to modify the floorplan in a way that helps improving its associated metric. For example, the temperature plugin adds a strategy that systematically moves power hungry blocks closer to the heat sink. During registration, each move is assigned with a probability of being used. The role of the move selection object is to pick a move at random and apply it to the current solution.
3.1. TOOL ARCHITECTURE

3.1.1 Elements subject to parameterization

Besides the evaluation engine, all the elements involved in the optimization process, namely the move selection, the cost function and the annealing schedule, are subject to parameterization and have an effect on the convergence of the optimization and the quality of the floorplan.

Cost function

The evaluation engine returns a set of metrics \( f \) that have to be simultaneously optimized. Such a multiobjective problem is stated as\(^1\):

\[
\min_X f = [f_1, f_2, ..., f_n]^T
\]  

(3.1)

A solution that can simultaneously optimize every objective rarely exists. Thus, a trade-off has to be made, prioritizing certain objectives over others. The most common manner to express this preference is to combine the objectives in a cost (or merit) function using a weighted sum:

\[
\min_X \text{cost}(f) = \sum_{i=1}^{n} w_i \cdot f_i
\]  

(3.2)

Then, the problem becomes the minimization of a single objective and solutions can be directly compared as having a higher or lower cost. However, determining the appropriate weights that will lead to a solution presenting the expected trade-off is a subjective task.

The difficulty lies in the fact that the relative importance of each weight depends on the circuit under design. For example, for some circuits it may be easy to find solutions that are both compact and have short wirelength. In such cases, the outcome of the optimization is only weakly sensitive to the value of the weights. On the other hand, if compact area and short wirelength are incompatible due to a difficult geometry and connectivity of the blocks, then the choice of the weights becomes crucial to pull the solution towards one objective or the other.

\(^1\)A minimization problem is considered without loss of generality. For maximization, the objectives are multiplied by \((-1)\).
Move selection

Compared to a random search, the use of heuristic moves increase the chances of improving the solution after each modification [AM03]. This can greatly improve the performance of the stochastic optimization, both in terms of runtime and of predictability of the result. However, with multiple moves available and with each one favoring a different objective, there must be a balance in their usage. Finding the ideal balance can only come from empirical experience. What the mofp framework offers is an API to declare the moves and assign selection probabilities to each.

In 3D partitioning, the blocks of a system can be moved inside a tier or between tiers. Intralayer moves correspond to classical 2D floorplanning and have the objective of compacting every layer. On the other hand, interlayer moves explore the possibility of placing densely connected blocks one on top of the other.

Practical experience has shown that interlayer moves cause the greatest perturbations in the floorplan and, when using the SA heuristic, they are increasingly unlikely to be accepted as the algorithm progresses [CWZ04]. Taking into account these concepts, the move selection mechanism in the mofp framework is composed of:

**Separate registers for the 2D and 3D moves:** The registers are fed by the plugins, which also assign a probability to each move. When all moves have been declared, the probabilities are normalized to add up one.

**A law restricting the use of 3D moves as the algorithm progresses:** The probability of applying a 3D move is restricted according to the piecewise linear function shown in Figure 3.3, with a linear decrease and a lower saturation point.

**Simulated Annealing schedule**

The last component influencing the optimization is the annealing schedule. The schedule controls how the probability of accepting uphill moves decreases as the optimization progresses. It, thus, controls the balance between exploration and exploitation of the design space. The literature proposes a
3.1. TOOL ARCHITECTURE

Figure 3.3: The probability of performing a 3D move decreases at each iteration. The saturation point $sat$ is defined as a percentage of the maximum number of iterations, $max$.

Multitude of implementations [Egl90], some being more or less complex. In this work, the choice for an annealing schedule was made on the basis of simplicity (i.e. few control parameters) and robustness. The statistical schedule proposed by Aarts and Laarhoven [AL85] was adopted based on the comparative study of Cohn and Fielding [CF99]. Its implementation in the mofp framework has the following characteristics:

**Temperature update function:** The temperature is kept constant for a certain number of moves and the next temperature is given by the function:

$$T_{k+1} = \frac{T_k}{1 + \frac{T_0 ln(1 + \lambda)}{3\sigma_k}}$$

(3.3)

where $\sigma_k$ is the standard deviation of the cost values observed during the $k_{th}$ temperature floor, and $\lambda$ is a tuning parameter.

**Moves per temperature floor** ($inner_{iter}$): A large number of moves at the same temperature floor does not help convergence, while a low number can present a noisy measure of $\sigma_k$ and affect the cooling schedule. In this implementation, $inner_{iter}$ is directly proportional to the number of blocks ($n_{blocks}$) by the parameter $k$:

$$inner_{iter} = k \cdot n_{blocks}$$

(3.4)

**Initial temperature:** a suitable initial temperature $T_0$ is one that results in a high probability $\chi_0$ of accepting uphill moves, allowing a large exploration of the neighborhood of $X_0$ (initial solution) in the early iterations. Some
authors suggest this probability to be around 0.8 [CF99]. The value of $T_0$ that satisfies such criteria varies according to the magnitude of the cost function and, consequently, to the circuit under design. To eliminate this dependency, it is possible to estimate $T_0$ in a first step. Such an estimation is performed during the first inner iterations by accepting all uphill moves and calculating the average cost increase $\Delta C^+_{\text{Avg}}$. Then, considering that a Boltzmann distribution is used in the annealer, $T_0$ is given by:

$$T_0 = -\Delta C^+_{\text{Avg}}/\ln(\chi_0)$$  \hspace{1cm} (3.5)

In the described implementation, only two parameters control the annealing schedule: the $\lambda$ parameter in the Aarts' schedule and the factor $k$ defining the number of moves executed at each temperature step.

### 3.2 A meta-optimization approach for parameter tuning

The previous section identified the tuning parameters of each element impacting the optimization process. Finding the right values for these parameters, i.e., values that yield good results for most circuits, is a time-consuming task requiring a good amount of empirical data and intuition. This section presents a method to automate this task by formulating it as another optimization problem (hence the term meta-optimization). The method is based on genetic algorithms (GA) and the concept of multi-scenario optimization.

Multi-scenario optimization is a response to the problem of making decisions in an uncertain environment and has applications in many financial and engineering problems [CC04, RYP+12]. The aim is to find a robust solution that works well for all possible conditions of the environment based on the analysis of a limited number of scenarios. In a multi-scenario approach, the multiobjective problem (3.1) becomes:

$$\min_{x} f = [f_{11}, f_{21}, \ldots, f_{n1}, \ldots, f_{1m}, f_{2m}, \ldots, f_{nm}]^T$$ \hspace{1cm} (3.6)

In the context of the present method, a scenario is a circuit, the decision
variable $\mathbf{x}$ is the vector of tuning parameters and the outcome $\mathbf{f}$ is the set of quality metrics returned by the floorplanning tool (Figure 3.4). To guarantee the robustness of the solution, the set of scenarios is composed by selecting a few known benchmarks, each stressing the floorplanner in a different way. Moreover, the meta-optimization problem also has the following characteristics:

**The outcome is discontinuous**: the floorplanning problem being combinatorial, a finite number of solutions exist.

**The outcome is noisy**: the floorplan solution depends on an initial guess and is obtained with a stochastic process.

**No prior knowledge is assumed**: neither of the value of the parameters (input), neither of the achievable trade-offs (output).

Genetic algorithms are well suited to solve such problems.

### 3.2.1 Pareto optimality and multiobjective GA

Genetic algorithms are stochastic global search methods that mimic natural evolution. They act over a population of potential solutions, applying intensification (crossover) and diversification (mutation) operators to explore the design space. The fittest individuals are selected and give birth to a new population, in the hope of improving the solution quality.

The selection of the best individuals is based on the criteria of Pareto-dominance. A solution $\mathbf{f}^a = [f_1^a \ldots f_n^a]$ *dominates* a solution $\mathbf{f}^b = [f_1^b \ldots f_n^b]$,

\[
C(S) = a + \ldots + a*WL
\]
A solution is said to be Pareto-optimal if it is not dominated by any other in the solution space $\mathcal{F}$. The set of all nondominated solutions form the so-called Pareto-frontier ($\partial\mathcal{F}^\prec$), representing the trade-off limits for a multiobjective problem.

Figure 3.5a illustrates this concept in the context of the meta-optimization problem for a single scenario. The frontier $\partial\mathcal{F}^\prec$ is bounded by the minimum of each individual objective. An intermediate solution in the front represents a compromise between the objectives. Another concept is that of the (unachievable) utopia point, simultaneously combining the best of both objectives.

Figure 3.5b transposes these concepts for the case of multiple scenarios. The primary objectives (area and wirelength) are repeated for each scenario yielding a 4-dimensional front. In this illustration, a rectangle which is well centered in the plot is a sign of good balance between the scenarios.
3.2. A META-OPTIMIZATION APPROACH FOR PARAMETER TUNING

Ranking solutions using Pareto-dominance eliminates the need for weighting the objectives in a scalar merit function. Therefore little prior knowledge about the problem is required and decisions are postponed to the end of the optimization, when the Pareto-front is returned.

3.2.2 Problem formulation

The decision vector \( \mathbf{x} = [x_1, \ldots, x_k] \) in (eq. 3.6) groups the elements mentioned in section 3.1.1 (Figure 3.6). All the parameters \( [x_1, \ldots, x_k] \) assume real values and are subject to lower and upper bounds. Table 3.1 details the conditions that apply to each variable. The number of variables related to cost function weights and moves selection depends on the problem in hand. The constraint on the variable \( P_f \) guarantees the shape of the PWL function in Figure 3.3. The empirical bounds on the shape of the PWL function impose few restrictions. However, some expertise is necessary to reduce the design space for the annealing schedule. The bounds on \( \lambda \) come from the literature and those for \( \text{inner}_{\text{iter}} \) are a guess made around the annealing schedule used in Parquet [AM03] (different from the one used in this work).

A population of individuals is evolved with crossover, mutation and selection operations. The present method uses the NSGA-II algorithm to select parents based on the crowding distance measure [DPAM02]. The selected parents are combined and modified with the usual operators:

- Arithmetic crossover: \( \mathbf{c} = \alpha \cdot \mathbf{a} + (1 - \alpha) \cdot \mathbf{b} \), with \( \alpha = U(0, 1) \)
- Gaussian mutation: \( \mathbf{c} = \mathbf{a} \cdot [1 + \mathcal{N}(0, 1)] \) \hspace{1cm} (3.8)

After any of these operations, violations of bound conditions are corrected. The probabilities for move selection \( (p_i, q_j) \) are normalized to add up 1 and a saturation operator \( \min/\max \) is applied to the remaining variables.
Table 3.1: Detailed list of variables for the meta-optimization

<table>
<thead>
<tr>
<th>Group/Variables</th>
<th>Bounds</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost function weights $w_1, w_2, \ldots, w_n$</td>
<td>$[0,1]$</td>
<td></td>
</tr>
<tr>
<td>Annealing schedule $\lambda$</td>
<td>$[0.3, 1.0]$</td>
<td>empirical bounds [CF99]</td>
</tr>
<tr>
<td>inner $\lambda$</td>
<td>$[6, 10]$</td>
<td>empirical bounds</td>
</tr>
<tr>
<td>2D moves selection $p_1, p_2, \ldots, p_m$</td>
<td>$[0, 1]$</td>
<td>constraint: $\sum_i^{m} p_i = 1$</td>
</tr>
<tr>
<td>3D moves selection $r_1, r_2, \ldots, r_k$</td>
<td>$[0, 1]$</td>
<td>constraint: $\sum_i^{k} r_i = 1$</td>
</tr>
<tr>
<td>3D/2D probability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_i$</td>
<td>$[0.1, 1]$</td>
<td>empirical bounds</td>
</tr>
<tr>
<td>$P_f$</td>
<td>$[0, 0.9]$</td>
<td>constraint: $P_f \leq P_i$</td>
</tr>
<tr>
<td>sat</td>
<td>$[0.5, 1]$</td>
<td>empirical bounds</td>
</tr>
</tbody>
</table>

As previously mentioned, the outcome of the floorplan evaluation is noisy. For a given vector $x$ used to parameterize the tool, the measured performance (e.g. area, wirelength, temperature) varies depending on the initial floorplan. To remove this dependency the measured performance should be the average of multiple runs. The choice of the number of runs is mainly limited by execution runtime.

3.2.3 Incremental optimization and validation

The ideas developed in the previous sections show that the meta-optimization problem is complex. It involves choices in the GA implementation, the bounds of the variables, the selection of the scenarios and the measurement of the objectives. To increase the chances of success, the problem can be solved by parts with an incremental approach.

The 3D floorplan problem is an extension of the 2D one. Hence, a floorplanner must already present a good performance in 2D cases to ensure good performance in 3D. The primary problem of area-wirelength minimization in 2D is solved first, eliminating all the variables related to inter-layer moves. The results of this step can be validated by testing extra scenarios. Then more complex problems can be addressed (e.g. 3D, temperature). The experience gained at each step is used to close the bounds on some variables, limiting the degrees of freedom for the next step.
4 MODULARITY

The mofp tool follows a modular approach, providing APIs to extend the core functionality. To account for different aspects of 3D design, modules can add:

- Evaluation engines to quantify these aspects
- Cost functions adapted to the target problems
- Heuristic strategies to speed-up the optimization

This chapter presents the modules developed in this work, going from the basic area-wirelength problem up to the joint optimization of area, temperature and application performance in a multicore system connected by a 3D Network-on-Chip.

4.1 Area-Wirelength problem in 3D

The area-wirelength module has a special role in the framework. This is because it solves the primary problem of floorplanning, and secondly because it imposes the floorplan representation to be used. We discuss its implementation, highlighting differences to commonly adopted formulations.

4.1.1 Area evaluation

The evaluation of area involves translating the topological representation to the \((x, y, z)\) coordinates of the blocks. In this work, a 3D version of the sequence pair (SP) representation is used.

SP is a general floorplan representation proposed by Murata et al. [MFNK96]. It consists of two permutation lists of \(n\) elements, where \(n\) is the number of blocks in the system. The two topological relations (left of or above) are encoded by the order in which the blocks appear in the two lists:
The graphical interpretation of the SP is exemplified in Figure 4.1, where its direct relationship to a constraint graph is shown. The computation of the \((x, y)\) coordinates of the blocks is efficiently done with the Longest Common Subsequence (LCS) packing algorithm proposed by Tang et al. [TTW00], which has worst case complexity \(O(n^2)\).

The extension of SP to represent a 3D floorplan (3D-SP) is straightforward, with one SP used for each layer. Then, interlayer moves consist of deleting a block in one SP and inserting it in another. A remark here is that, due to the quadratic complexity of the LCS algorithm, a 3D-SP is processed faster than the 2D case. Indeed, each layer in the 3D-SP will contain a fraction of the \(n\) blocks in the system. Then, by the Cauchy-Schwarz inequality:

\[
O(n^2) > O(m^2) + O(k^2), \text{ with } n = m + k
\]  

Therefore, arguments stating that SP cannot handle a large number of blocks [CCWW00] lose strength in the 3D case. Once the blocks’ coordinates have been calculated, the wirelength is estimated computing the \(L_1\) distance (called Manhattan or half-perimeter distance) between connected pins.

### 4.1.2 Accounting for TSV dimensions

Depending on the packaging technology (e.g. D2D, D2W), the TSV features will be different. The pitch between TSVs can range from a few microns up
4.1. AREA-WIRELENGTH PROBLEM IN 3D

The area overhead of the TSVs is distributed across the floorplan in the form of a guard ring around the blocks.

to hundreds of microns, and their depths vary according to the thinning of the wafer. Although these dimensions are reduced as technology matures, the impact of having thousands of TSVs in a design cannot be neglected and must be addressed during partitioning.

A precise measure of this overhead requires the use of some iterative algorithm that can place the TSVs under routing constraints and exploit eventual whitespaces in the floorplan ([XSXY10, HR07]). Such methods are commonly applied only to the result of the floorplanning step and not during it. Instead, during floorplanning, faster approximations can be used.

In this work the TSV dimensions are directly taken into account in the area and wirelength evaluation. The depth of each TSV is added to the total wirelength. For their area overhead, the suggestion of Cong et al. [CWZ04] is followed. It consists of enlarging the blocks according to the number of TSVs before evaluating their positions, so as to create a guard ring around them (Figure 4.2).

In mofp, this suggestion is implemented as follows: when any interlayer move is performed, the evaluation module updates, for each layer $l$, the total area of the blocks in it ($A_{\text{blocks},l}$) and of the number of TSVs traversing its substrate. Based on the pitch of the TSVs, their area overhead is calculated ($A_{\text{TSV},l}$). Then, an enlargement factor $\alpha_i$ is defined so as to distribute this overhead in proportion to the size of each block $b_i$ (width $w_i$ and length $l_i$):

$$\alpha_i.w_i \cdot \alpha_i.l_i = w_i.l_i \left(1 + \frac{A_{\text{TSV},l}}{A_{\text{blocks},l}}\right) \Rightarrow \alpha_i = \sqrt{1 + \frac{A_{\text{TSV},l}}{A_{\text{blocks},l}}} \quad (4.3)$$
4.1.3 Problem formulation

The considered problem is to partition a system into dies of the same area. The cost function is formulated as:

\[
\text{cost} = \alpha \cdot \text{Footprint} + \beta \cdot \Sigma \text{Area} + \gamma \cdot \text{Wire}
\] (4.4)

where \(\alpha\), \(\beta\) and \(\gamma\) are weight parameters; \(\text{Footprint}\) is the product of the largest width and the largest height; \(\Sigma \text{Area}\) is the sum of areas of all layers and \(\text{Wire}\) is the wirelength.

The introduction of the \(\Sigma \text{Area}\) goal comes from the observation that the footprint of a 3D system is defined by, at most, two different layers (Figure 4.3). Consequently, the benefit of a move that reduces the area of any smaller layer is not captured by the \(\text{Footprint}\) objective. Instead, it is captured by \(\Sigma \text{Area}\)\(^1\).

The normalization of each goal is as follows:

\[
\text{Footprint}_{\text{norm}} = \frac{\text{Footprint}_{\text{new}} \cdot N_{\text{layers}}}{A_{\text{blocks}}} \tag{4.5}
\]

\[
\Sigma \text{Area}_{\text{norm}} = \frac{\Sigma \text{Area}_{\text{new}}}{A_{\text{blocks}}} \tag{4.6}
\]

\[
\text{Wire}_{\text{norm}} = \frac{\text{Wire}_{\text{new}}}{\text{Wire}_{\text{old}}} \tag{4.7}
\]

\(^1\)A similar reasoning was used by Falkersten et al. [FXCW10], but with a formulation that forces an aspect ratio of 1.0.
4.1. AREA-WIRELENGTH PROBLEM IN 3D

Figure 4.4: The notions of wirelength minimization move and slack demonstrated on a 2D example (a) The cross indicates the location that minimizes the wirelength between blocks A,D and F (b) X and Y slack of block C. The single-headed arrows travel the X and Y critical paths.

The ΣArea and Footprint goals are normalized by absolute values corresponding to the minimum possible area (utopia). The Wire goal has to be normalized relative to the previous iteration since it is not generally known what the minimal value that can be achieved is and at what area expense.

Some authors have also proposed the minimization of the TSV count by adding a factor ψ ∗ NTSV to the cost function [CWZ04],[ZML+07]. However, in their approaches, the TSV dimensions are not taken into account during the floorplan, which makes the choice of the weight ψ quite subjective as it has to be adapted according to the dimensions of the system under design and the packaging technology. The proposed approach of adding the TSV overhead directly in the area and wirelength computation is, thus, more objective and eliminates an unnecessary term from the cost function.

4.1.4 Moves

The strategies for area-wirelength minimization implemented in mofp are largely based on those proposed by Adya and Markov [AM03], the authors of the open-source tool Parquet. Among the important concepts developed by these authors are the notion of slack in a SP floorplan and the use of wirelength minimization moves.

A wirelength minimization move (Figure 4.4a) consists of moving a block bi close to the “ideal” location that would minimize the wirelength of its incident nets. This “ideal” location (xa, ya) is simply the barycenter of all
modules connected to $b_i$, weighted by the net degree. Once a block $b_j$, sitting close to $(x_a, y_a)$ is identified, $b_i$ is moved next to $b_j$ in the SP.

The notion of slack in a floorplan is analogous to that in Static Timing Analysis (STA). It is the distance that a block can be moved in a given direction without changing the floorplan outline. As in STA, there is also the notion of a critical path, which constrains the floorplan dimensions (Figure 4.4b). A “slack” move takes a block from the critical path and places it next to a block with large slack.

In this work, the moves implemented in Parquet-4.5 are modified and adapted to the 3D case. The solution can then be perturbed in the following ways:

- **3D moves:**
  1. Swap blocks from two layers
  2. Move block to another layer
  3. Slack move between two layers
  4. Wirelength minimization move between layers
- **2D moves:**
  1. Random permutation of the SP
  2. Block rotation
  3. Wirelength minimization move
  4. Slack move
  5. Slack move with block rotation

4.2 Thermal-aware partitioning

Thermal dissipation is a critical design aspect in 3D-ICs and must be considered, along with area and wirelength, as a criteria for partitioning. 3D thermal-aware floorplanners have been studied by several authors [CWZ04, ZML+07, XSXY10]. In the presentation of this module, we comment on the common approaches and propose a new formulation of the problem that breaks the simulated annealing process into two sequential phases.
4.2. THERMAL-AWARE PARTITIONING

4.2.1 Temperature evaluation

The evaluation module counts two implementations. One performs a detailed analysis, suitable for verification. The other uses a simplified approach that executes faster, allowing it to be embedded in the optimization loop.

For the detailed analysis, the popular thermal simulator Hotspot [SSH+03] is used. Hotspot models the temperature in the chip as a linear system using the thermal-electrical analogy (Figure 4.5a):

\[
V = R \cdot I \leftrightarrow T = R_{th} \cdot P
\]

\[
\text{voltage} \leftrightarrow \text{temperature} \\
\text{current} \leftrightarrow \text{power} \\
\text{resistance} \leftrightarrow \text{thermal resistance}
\]

(4.8)

The input to the tool is a description of the stack, indicating the thickness and thermal resistivity of each layer. The designer is free to include additional passive layers (e.g. metalization and bonding) to better model the thermal behavior of the chip (Figure 4.6a). The total of L layers is then meshed into NxNxL cells, i.e. the area is divided into N rows and N columns, the thickness of a cell is the thickness of the corresponding layer. Cells are connected to their six neighbors with thermal resistances, and are assigned with a power value according to the layout of the blocks in the floorplan (Figure 4.6b). Power is assumed to be homogeneous within a block.

The resulting linear system \( T = R_{th} \cdot P \) is solved using a multi-grid method. The full NxNxL linear system is coarsened down to a 1x1xL grid, where an initial solution is easily computed and then successively refined back us-
Figure 4.6: From the physical structure of the stack to the resistive models.

The choice of the grid size N (number of rows and columns) is different between the detailed and the tile analysis. In the detailed analysis, a fine grid is desired to accurately model local effects at the interfaces between blocks. In tile analysis, this is not the case as tiles are independent of their neighbors. With the assumption that the power density is homogeneous within a block, any tile covering an area smaller than the smallest block presents no interest. On the other hand, having tiles that encompass more than one block is a manner to account for part of the lateral heat spread. Figure 4.7 shows a comparison between the detailed and the tile analysis for different grid sizes. The floorplan on the left is colored according to the

...
4.2. THERMAL-AWARE PARTITIONING

**Figure 4.7:** The effect of the grid size in tile analysis.

**Figure 4.8:** The number of cells in the grid is set according to the dimensions of the smallest block. For square grids, \( n_{\text{rows}} \times n_{\text{cols}} \) can be computed using the area of the blocks and ignoring their aspect ratio.

blocks’ power. It is noticeable that, with a fine grid, the result of the tile analysis is an image of the power map, while a coarser grid correlates better to the detailed analysis.

Thus, the designer inputs two distinct grid size values to \texttt{mofp}, one for optimization (tile analysis) and another for verification (detailed analysis). The reasoning presented in Figure 4.8 can be used to estimate the number of rows and columns for each grid. The direct reuse of \texttt{Hotspot} routines in \texttt{mofp} entails that the grid is kept the same during the optimization, even though the footprint of the chip varies at each iteration. Consequently, the area of the tiles changes across the iterations. In the first iterations (large footprint, large tiles), the tile analysis can underestimates the thermal profile of the chip. This effect disappears when the floorplan becomes compact.

Thesis on Micro et Nanotechnologies
Ecole Centrale Lyon

Felipe Frantz Ferreira / 2012
4.2.2 Problem formulation - A two-phase optimization

The common approach for thermal-aware floorplanning is to have a cost function minimizing the maximum temperature [CWZ04][ZML+07]:

\[ C(S) = \alpha \cdot \text{Area} + \beta \cdot \text{Wire} + \gamma \cdot T_{\text{max}} \] (4.9)

The choice of the weights in this problem is difficult to balance. While area and wirelength minimization can coexist, area and temperature are diametrically opposed objectives, i.e. temperature is always reduced if the blocks are moved away from each other. This is not a problem in itself, in the sense of optimization. The difficulty emerges from the floorplanner starting with an unorganized floorplan and moving to a compact one. With a larger footprint, the initial solution presents low temperature and impedes the optimizer from moving to solutions with competitive area\(^2\). The difficulty in balancing the weights of the cost function has been reported by Xiao et al. [XSXY10]. In that work the authors proposed a two-phase algorithm: optimizing \textit{area-wirelength} first phase and \textit{area-temperature} in the second.

In this work, an algorithm with two phases (\(\phi_1, \phi_2\)) is also proposed. The differences are the following:

- Power distribution is optimized in \(\phi_1\);
- A formulation accounting for thermal gradients is used in \(\phi_2\);
- A single annealing schedule is used and the transition from \(\phi_1\) to \(\phi_2\) is based on the area.

Phase 1 - Thermally-efficient power density distribution

From the 1D approximation of the vertical heat flow on the chip (Figure 4.9.a) it can be noticed that a power distribution with a pyramidal shape (Figure 4.9.b) will implicitly reduce peak temperature. Phase 1 seeks to arrange the blocks in \(n\) layers so that the more power-hungry are placed closer to the heat sink.

\(^2\)This effect is more noticeable with the Sequence-Pair representation, in which a randomly generated floorplan can be extremely unorganized in comparison to one generated using a compact representation such as B*-Tree.
4.2. THERMAL-AWARE PARTITIONING

\[
T_k = \sum_{i=1}^{k} (P_i \sum_{j=1}^{i} R_j)
\]

\[
P_Dens = \sum_{i=1}^{n} \frac{P_i}{\text{Area}} \cdot q_i \quad (4.11)
\]

\[
q_i = \frac{R_1}{\sum_{j=1}^{i} R_j} \quad (4.12)
\]

Figure 4.9: (a) Vertical heat flow model (b) Shape of a suitable power distribution that minimizes temperature.

Figure 4.10: (a) The proposed two phase algorithm and the switching criteria and (b) a thermal profile along a cross section of the \(xy\) plane.

The cost function expressing this objective is formulated as:

\[
cost = \alpha \cdot \text{Area} + \beta \cdot \text{Wire} + \delta \cdot \frac{1}{P_{Dens}} \quad (4.10)
\]

where \(P_{Dens}\) is a weighted sum of the power density of each layer defined as:

The formulation of the weighting factors \(q_i\) is arbitrary, as long as their values decrease for higher layers, guiding the algorithm to the pyramidal power distribution. Here they have been related to the thermal resistances in the vertical model and normalized to the range \((0,1]\).

With such a formulation, the maximization of the power density has a dual impact, reducing area and temperature at the same time.
Phase 2 - Minimize thermal gradients  

The second phase involves thermal simulations (tile analysis) and starts once the area reaches a threshold $\lambda$ that ensures that all layers are occupied. To guarantee this requirement, the threshold must satisfy $\lambda \leq A_{2D}/(n-1)$ where $A_{2D}$ is the area of the design in a 2D configuration and $n$ is the number of available layers (Figure 4.10a).

During $\phi_2$, the cost function (4.10) is augmented with the temperature:

$$C(S) = \alpha \cdot \text{Area} + \beta \cdot \text{Wire} + \delta \cdot \frac{1}{P_{\text{Dens}}} + \gamma \cdot \Sigma T_{\text{max}} \quad (4.13)$$

Instead of minimizing only the maximal temperature, the sum of peak temperatures on each layer is considered. This formulation is motivated by the observation that the $(x,y)$ coordinates of the hottest spot in each layer do not always coincide (as exemplified in Figure 4.10b). When these situations occur, this formulation is able to reward the improvements in the thermal profile of each layer (i.e. gradient reduction).

4.2.3 Moves - The power density heuristic

This module does not introduce any new moves other than those previously presented for the area-wirelength problem. Instead, it adds a filter to the mechanism that randomly picks the next block to be moved. The aim of this filter is to systematically move power hungry blocks towards the heat sink. It imposes that, in inter-layer moves, the selection of a block is subject to its power density and to the destination layer. If a block is being moved to a lower (resp. upper) layer, it must satisfy the condition of having a power density greater (resp. less) than the average power density of the blocks in its current layer. Blocks are selected randomly until this condition is satisfied, at which point the inter-layer move continues its normal execution.

4.3 MoNICA - Traffic optimization for 3D NoC-based MPSoCs

The optimization strategies developed in the previous modules have considered that communicating blocks are connected through dedicated wires.
4.3. MONICA - TRAFFIC OPTIMIZATION FOR 3D NOC-BASED MPSOCs

Figure 4.11: NoC-based MPSoC: cores are connected in a network, sharing the communication resources. (courtesy D.Matos)

However, this paradigm does not scale with the growing number of cores\(^3\) in today's MPSoCs. To attain flexibility and scalability requirements, the communication resources have to be intelligently shared. To this point, the NoC paradigm provides the solution to separate the application functionality (messages exchanged between cores) from the physical interconnect structure (a network of routers wired together, see Figure 4.11).

To include the NoC paradigm in the partitioning tool, we have to account for this separation and treat the problem from two perspectives:

- **Application mapping**: assign cores to network nodes in order to optimize the traffic of messages;

- **Chip floorplanning**: account for the application mapping and the topology of the network when optimizing the placement.

Some authors have proposed to treat each point separately, in a sequential approach. We propose to treat them simultaneously using a nested optimization scheme. This nested scheme couples the problems but separates the optimization strategies, allowing us to treat them efficiently.

This module has been developed in collaboration with researchers from CEA-Leti (France) and UFRGS (Brazil), who provided the tools to route and simulate the application traffic in a NoC-based MPSoC. The result of the integration of mofp with these tools is called MoNICA (Multiobjective Network Interconnect Chip Architect).

\(^{3}\)In the study of NoCs the term core is used in the place of block.
4.3.1 Problem formulation - A nested optimization process

To account for the characteristics of this problem, the original description of a set of blocks connected by nets is substituted by the following elements:

**MPSoC description:** a set of $k$ cores $C = \{c_1 \ldots c_k\}$, each core being defined by its dimensions and average power consumption;

**Traffic pattern:** the interactions between cores for a specific application. These are messages defined by a source $c_i$, a destination $c_j$ and the required bandwidth;

**3D-NoC topology:** a set of $k$ network nodes $N = \{n_1 \ldots n_k\}$ and their connectivity (e.g. mesh, torus). The connectivity is both planar and vertical, defining how the nodes are partitioned across layers of the 3D chip;

![Figure 4.12: Features considered by MoNICA (courtesy D.Matos)](image)

Considering the two perspectives of the problem, we formulate two problems:

**Mapping problem:** a mapping assigns each core to a network node (Figure 4.12). The quality of a mapping is assessed by simulating the traffic on the network. A good mapping allows messages to be routed through short paths, minimizing latency and congestion of the network.

**Floorplan problem:** the mapping and the NoC topology define the partitioning of the cores over the chip and their connectivity. Considering this and the different shapes the cores (Figure 4.13), the floorplan problem is to lay out the cores so that area and temperature are minimized.

The two problems are tightly coupled: the adopted NoC topology (e.g. mesh, torus) imposes the main guidelines for the chip floorplan and modifications to the mapping correspond to changes in the physical position of the cores,
4.3. MONICA - TRAFFIC OPTIMIZATION FOR 3D NOC-BASED 
MPSOCs

Figure 4.13: Translating a mapping to a floorplan. The dimensions of the blocks are considered and their connectivity is set according to the network topology.

requiring the computation of a new floorplan. Taking this coupling into account, an efficient way to perform a joint optimization of the communication, temperature and area is to break the optimization into two levels:

1. A main loop that manages the mapping: it moves cores to different layers and positions in the network and is, thus, responsible for the major changes in the floorplan and thermal profile of the chip.
2. A nested loop that refines the floorplan after each change to the mapping.

This gives an efficient structure to build a NoC-aware 3D floorplanner (Figure 4.14). Information about the NoC topology and the operations to modify the mapping are only encoded in the main loop. The nested loop then refines the floorplan of each layer and evaluates the thermal distribution of the chip (tile analysis). In parallel, the mapping is used to calculate the routes for the application traffic and provide a quick approximation of the latency. Once the optimization terminates, the results are fed to HotSpot and to a SystemC simulator for a fine grain analysis of temperature and latency.

Like the floorplan, the mapping is also treated as a combinatorial optimization problem. Therefore, no large modifications were required to integrate it into the mofp framework. The specific strategies implemented for each problem are discussed next.

4.3.2 NoC-aware floorplan

The main optimization loop groups the three concerns of area, temperature and latency minimization in its cost function:

\[
Cost = \alpha \cdot Footprint + \beta \cdot \Sigma T_{max} + \gamma \cdot Latency
\] (4.14)
The optimization variable at this step is the core-to-node mapping. The mapping is represented like the diagram in Figure 4.11: i.e. a 3D matrix with $N$ positions (the nodes), to which the cores are assigned. The mapping is changed by permuting the assignment of the cores. The shape of the cores is not considered here. However, the actions of changing the positions of the cores inside a layer and between layers accounts for a great part of the 3D floorplan problem and have great impact on the thermal profile of the chip and the latency of the network. Thus, we exploit the information about the application graph and the power of the blocks to modify the mapping. Based on the work done for the previous modules, three operations are allowed:

1. Random permutation: both inter- and intralayers;
2. Random power move: pick a core with high (resp. low) power consumption and move it to a layer closer (resp. farther) from the heat sink.
3. Latency move: analyze the network traffic and identify messages with large latency. Pick a core emitting one of these messages and place it next to the destination (similar to the wirelength move in section 4.1).

As previously stated, during optimization, the latency of the network is approximated by metrics returned by the 3D routing algorithm developed by our partners. The metrics considered are hops and occupation. Hops is the number of routers that a message traverses from source to destination, while occupation is the number of messages that traverse a router. Considering the
4.3. MONICA - TRAFFIC OPTIMIZATION FOR 3D NOC-BASED MPSOCs

entire system, for the purpose of optimization, the latency is approximated by:

\[ \text{Latency} \approx \text{average}(\text{Hops}) + \text{average}(\text{Occupation}) \]  \hspace{1cm} (4.15)

This formulation rewards mappings that put communicating cores closer and that avoid occupying the same routing resource (which would cause traffic congestion).

At the present state of this work, MoNICA considers a homogeneous 3D mesh topology for the network, with each layer having the same number of nodes. This, however, does not limit the approach. Support to arbitrary topologies can be added by specializing the operations that modify the mapping and by providing a suitable routing algorithm.

![Optimal floorplan Permutation in the mapping Modified floorplan](image)

**Figure 4.15:** The effect of a modification in the mapping on the optimal floorplan.

4.3.3 Floorplan refinement

After each modification in the mapping, the floorplan needs to be re-optimized to account for the permutation of the core positions and rearrange them into the most compact layout (Figure 4.15). This floorplan refinement process considers all layers simultaneously. At each iteration, one layer is selected and its floorplan modified. Besides the individual area of the layers (\( \Sigma \text{Area} \)), the cost function also accounts for the wirelength as a means to preserve the overall topology of the network:

\[ \text{Cost} = \alpha \cdot \Sigma \text{Area} + \beta \cdot \text{Wire}_{2D} + \gamma \cdot \text{Wire}_{3D} \]  \hspace{1cm} (4.16)

The computation of the wirelength is divided into two terms: \( \text{Wire}_{2D} \) measures the half-perimeter distance between routers in the plane, whereas \( \text{Wire}_{3D} \) performs the same measure considering the vertical connectivity, providing a measure of the vertical alignment of the routers. By splitting
the wirelength measure into these two metrics, we can assign a higher value to $\gamma$ to prioritize vertical alignment over compact area. Temperature is not considered here, because the refinement process does not modify the mapping, and therefore has no significant margins to change the thermal profile of the chip.

To speed up the process, we take advantage of the regularity of the NoC to generate a near-optimal initial floorplan, instead of a random one. For a $n \times m$ mesh structure, a good initial solution is to align the cores in rows and columns. Figure 4.16 shows how this grid layout can be captured in a SP representation. The diagram highlights the relations of core B, which is above C and F, and the arrows indicate how to order the elements in the lists. The initial SP for a grid layout is, thus, built in the general form:

$$\langle \langle \text{col}_1, \text{col}_2, \ldots, \text{col}_m \rangle, \langle \text{row}_n, \text{row}_{n-1}, \ldots, \text{row}_1 \rangle \rangle \quad (4.17)$$

With such initialization, only small and localized modifications need to be applied to the floorplan. Only two operations are implemented:

1. Rotation;
2. Random permutation of elements of only one of the sequences at a time and limited to the $m$ or $n$ closest neighbors (depending on which list is being modified).

This localized search allows finding a compact solution with few iterations. This is particularly important in a nested optimization scheme. Moreover, the floorplan is initialized only once. Subsequent calls to the refinement process reuse the floorplan of the previous mapping, optimizing it in an incremental fashion.

![Grid floorplan and Sequence Pair diagram](image)

**Figure 4.16:** Blocks laid out in a grid and the equivalent SP diagram
4.4 Summary

This chapter presented the additions that enable mofp to deal with different aspects of the 3D partitioning problem. The modules developed in this work were presented in an incremental order and put in the perspective of the state of the art to clarify our contributions. The area problem was treated first, fixing the choice for the floorplan representation used throughout this work (Sequence Pair).

A second module integrates the thermal dissipation aspect, a major issue of 3D-IC technology. Due to the difficulties in trading off area and temperature, we proposed a two-phase algorithm. In the first phase, a weighted sum of the power density of each layer is used as a metric to implicitly minimize temperature. Once the chip area becomes sufficiently compact, the algorithm switches to its second phase. At this point thermal simulations are executed and a temperature term is added to the cost function of the problem.

Finally, in a collaboration with other institutions, we treated the communication infrastructure of 3D-ICs and developed MoNICA, a partitioning flow for NoC-based MPSoCs built on top of the two previous modules. Using a nested optimization scheme, MoNICA decomposes the problem in a hierarchical approach: the main optimization loop maps cores to network nodes to optimize the traffic from a functional point of view. After each modification to the functional mapping, a nested loop performs the optimal placement of the cores.
5 Experimental Results

A design automation tool is validated through its application. To this end we have performed a series of experiments on several benchmarks to assess the efficiency of the strategies we propose. Due to the stochastic nature of the optimization process in mofp, the analysis is based on statistical results, collected over multiple runs of the tool.

5.1 Implementation and test environment

mofp is implemented in C++. It makes extensive use of the Boost library\(^1\) and uses MathGL\(^2\) to produce animations and interactive plots of the 3D chip. All experiments performed next were executed on an Intel Xeon machine running Linux (4 CPU, 8GB RAM, 2.4GHz).

5.2 Automatic parameter tuning with GA

The experimental study is developed around five benchmark circuits: ami\(_{33}\), ami\(_{49}\) from MCNC\(^3\) and n\(_{100}\), n\(_{200}\) and n\(_{300}\) from GSRC\(^4\).

5.2.1 Configuring the meta-optimization problem

The first step in the meta-optimization flow is to select, among the five benchmarks circuits, those that will compose the training set and those that will serve for validation. It is important for the training set to be rich, stressing the tool in different ways. Figure 5.1 shows sample floorplans for the benchmarks, while Figure 5.2 compares the dispersion in their blocks’

\(^1\)http://www.boost.org/
\(^2\)http://mathgl.sourceforge.net/
\(^3\)Microelectronics Center of North Carolina - http://www.mcnc.org
\(^4\)Gigascale Systems Research Center - http://www.gigascale.org
5.2. AUTOMATIC PARAMETER TUNING WITH GA

Figure 5.1: 2D floorplan solutions for the considered benchmarks.

Figure 5.2: Characteristics of the studied benchmarks.

dimensions. Based on this data, it appears that the MCNC benchmarks are most likely harder to place due to the larger dispersion in area and aspect ratio, while the GSRC family should stress the ability of the flooplanner to converge quickly even with many blocks. Thus, the employed training set uses the ami49 and n100 circuits.

The multi-objective GA implemented in MATLAB was used [Mat]. The population was composed of 32 individuals initialized randomly and evolved for 60 generations. The fitness of the individuals was measured over 8 floopl plan runs. The evaluation step was parallelized with a shell script, speeding up the resolution by a factor 4 (number of available CPUs).

The experiments here target the area-wirelength problem. The incremental approach was adopted, separating the 2D and 3D problems. Table 5.1 shows the number of variables involved in each step. The solution from the first step is inserted in the initial population of the second step, propagating the values found for annealing schedule and cost weights. The 2D moves, on the other hand, are fixed to their optimal values from the first step. With this experimental setup, the total meta-optimization runtime (2D plus 3D steps) is around 40 hours.
Table 5.1: Meta-optimization variables

<table>
<thead>
<tr>
<th>Algorithm structure</th>
<th>No. of variables</th>
<th>Optimization step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annealing schedule</td>
<td>2</td>
<td>1,2</td>
</tr>
<tr>
<td>Cost function weights</td>
<td>3</td>
<td>1,2</td>
</tr>
<tr>
<td>2D moves selection</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3D moves selection</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Probability 2D/3D</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 5.3: Results of the meta-optimization. On the left, the 4-Dimensional Pareto Front. The values are normalized by the average of each metric. The dashed rectangle indicates the retained parameter set, which is detailed on the right.

Figure 5.3 shows the obtained 4-dimensional Pareto front after the 3D step. A trade-off solution has been selected manually (outlined by the dashed rectangle). This solution becomes the default parameterization for the floor-planner.

5.2.2 Validation with benchmarks

The meta-optimization approach is validated comparing the solution quality of mofp to the state of the art. The aim of these experiments is not to compare the best solutions found for each benchmark, but the average results obtained with different initial solutions. While the best reported results vary little since the used benchmarks have been studied for years, a reduction in the average indicates a greater robustness of the tool.

The first experiment consists of a comparison with Parquet [AM03] in a 2D test case. Both tools were executed for the same number of iterations and the
5.2. AUTOMATIC PARAMETER TUNING WITH GA

Table 5.2: Performance comparison in 2D

<table>
<thead>
<tr>
<th>Circuit</th>
<th># Iter.</th>
<th>Area (WS) ((mm^2))</th>
<th>Wire Area (WS) ((\mu m))</th>
<th>Wire Area (WS) ((mm^2))</th>
<th>Wire Area (WS) ((\mu m))</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami33</td>
<td>66k</td>
<td>1.32 (14.1%)</td>
<td>59983</td>
<td>53801</td>
<td></td>
</tr>
<tr>
<td>ami49</td>
<td>98k</td>
<td>40.60 (14.5%)</td>
<td>800884</td>
<td>713063</td>
<td></td>
</tr>
<tr>
<td>n100</td>
<td>200k</td>
<td>0.197 (9.75%)</td>
<td>233973</td>
<td>233656</td>
<td></td>
</tr>
<tr>
<td>n200</td>
<td>400k</td>
<td>0.195 (11.0%)</td>
<td>441127</td>
<td>442236</td>
<td></td>
</tr>
<tr>
<td>n300</td>
<td>600k</td>
<td>0.304 (11.3%)</td>
<td>623822</td>
<td>621030</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.4: Comparisons between mofp and Parquet based on table 5.2

The results were averaged over 100 independent runs for each benchmark\(^5\). The results are detailed in Table 5.2 and illustrated in Figure 5.4. A whitespace\(^6\) reduction is observed in most benchmarks and a significant improvement in the average wirelength is obtained for ami33 and ami49 without compromising the other benchmarks. These results prove that the meta-optimization can successfully parameterize the floorplanner.

The performance in a 3D case with four layers is compared to the published results of 3D-STAF [ZML\(^+\)07] and the CBA method proposed by Cong et al. [CWZ04]. To enable comparison with these works, the objective of minimizing the number of vias is introduced in the mofp cost function (4.4) and its weight is manually configured to achieve similar via count. Results are reported in Table 5.3 and their analysis is supported with the illustration in Figure 5.4. First, the number of 3D-vias used by mofp is always intermediate with respect to the other tools. The results are then comparable and any

---

\(^5\)Parquet is executed using Sequence Pair and default weights. Terminals are placed at the boundary of the chip as defined in the benchmarks available at [http://vlsicad.eecs.umich.edu/BK/](http://vlsicad.eecs.umich.edu/BK/)

\(^6\)Whitespace is the proportion of the chip area that is not occupied by blocks, i.e: whitespace = \((\text{area}_{\text{chip}} - \text{area}_{\text{blocks}})/\text{area}_{\text{chip}}\). In a 3D case, the chip area is multiplied by the number of layers to calculate the whitespace.
Table 5.3: Footprint, wirelength and 3D-vias minimization for 4 layers stack.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Area (WS)</th>
<th>Wire</th>
<th>Vias</th>
<th>Time</th>
<th>Area (WS)</th>
<th>Wire</th>
<th>Vias</th>
<th>Time</th>
<th>Area (WS)</th>
<th>Wire</th>
<th>Vias</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami33</td>
<td>0.364 (25.9%)</td>
<td>26.1</td>
<td>116</td>
<td>2</td>
<td>0.379 (31.1%)</td>
<td>22.0</td>
<td>122</td>
<td>52</td>
<td>0.353 (22.1%)</td>
<td>22.5</td>
<td>93</td>
<td>23</td>
</tr>
<tr>
<td>ami49</td>
<td>11.48 (29.6%)</td>
<td>427.2</td>
<td>194</td>
<td>5</td>
<td>13.49 (52.2%)</td>
<td>437.5</td>
<td>227</td>
<td>57</td>
<td>14.90 (68.2%)</td>
<td>446.8</td>
<td>179</td>
<td>86</td>
</tr>
<tr>
<td>n100</td>
<td>0.050 (11.5%)</td>
<td>92.7</td>
<td>881</td>
<td>20</td>
<td>0.059 (31.5%)</td>
<td>91.3</td>
<td>828</td>
<td>68</td>
<td>0.053 (17.9%)</td>
<td>100.5</td>
<td>955</td>
<td>313</td>
</tr>
<tr>
<td>n200</td>
<td>0.048 (10.1%)</td>
<td>173.9</td>
<td>1810</td>
<td>84</td>
<td>0.059 (34.3%)</td>
<td>168.6</td>
<td>1729</td>
<td>397</td>
<td>0.058 (31.4%)</td>
<td>210.3</td>
<td>2093</td>
<td>1994</td>
</tr>
<tr>
<td>n300</td>
<td>0.075 (10.3%)</td>
<td>257.2</td>
<td>1914</td>
<td>160</td>
<td>0.097 (42.0%)</td>
<td>237.9</td>
<td>1554</td>
<td>392</td>
<td>0.089 (30.3%)</td>
<td>315</td>
<td>2326</td>
<td>3480</td>
</tr>
</tbody>
</table>

To allow comparison, terminals are placed at the center of the top-most layer. Results for this work are averaged over 100 independent runs and each run executes the same number of iterations reported in table 5.2. The data for CBA and 3D-STAF is presented as in the original publications for the area-wirelength minimization problem. The runtime values of CBA are not directly comparable.

Figure 5.5: Graphical comparison based on the results from table 5.3.
improvement in the other metrics is not a consequence of an overly intensive use of TSVs. The comparison of whitespace shows a clear advantage of mofp. Whitespace is significantly lower in all benchmarks and, most importantly, it does not worsen as the number of blocks increases. The wirelength is, in average, 5% longer than 3D-STAF, but still 8% smaller than CBA. Notably, the results of 3D-STAF favor wirelength over area, but the trade-off is debatable since die area is the main factor determining the economical cost of a chip. The important remark here is that, for a four layer stack, any design with whitespace larger than 30% could fit in three layers with the same footprint.

The runtime comparison in Figure 5.6 reveals the same trends reported by the authors of 3D-STAF. The scalability of mofp and CBA is essentially the same, as both are based on topological representations of identical computational complexity. While the reported results are not sufficient to confirm the scalability of 3D-STAF beyond 300 blocks, it is clear that below this number, the present work is more competitive. Moreover, the scalability problem can be addressed through hierarchical approaches that reduce the total number of blocks in the problem by clustering them [AM03, RAPM06, SEG +08].

5.2.3 Accounting for TSV dimensions

The experiment in this section shows the improvement that can be achieved if the TSV dimensions are included directly in the area and wirelength calculation as proposed in section 4.1.2. The following cases are considered:
CHAPTER 5. EXPERIMENTAL RESULTS

Figure 5.7: Minimizing via count (a) and then reserving area for vias (b) or accounting for via dimensions during the whole optimization process (c). Here, solution (c) uses more vias but has about 3% less area and wirelength than (b).

A posteriori insertion: The floorplan is obtained minimizing the TSV count (as in the previous experiment). The solution is then modified to reserve area for the TSVs and their depth is added to the wirelength. Figures 5.7a and 5.7b illustrate this approach.

A priori insertion: The cost function includes the TSV contribution to area and wirelength during the entire optimization process (see Figure 5.7c).

To account for the TSVs, a stack with all four layers in Face-to-Back (F2B) orientation and terminals on top is assumed. In this configuration, the bottom layer (attached to the heat sink) has no TSVs and the top layer (having the Cu contacts) is traversed by all vias that need to reach the terminals. Dimensions for TSVs are taken from ITRS projections [ITR09] and two cases are considered: for smaller benchmarks (n100, n200, n300) TSVs have pitch of 4µm x 4µm and depth of 10µm, dimensions compatible with Wafer-to-Wafer bonding. For ami33 and ami49, larger TSVs compatible with Die-to-Die and Die-to-Wafer are used with pitch of 8µm x 8µm and depth of 20µm.
5.3 Thermal-aware floorplanning

The following experiments perform temperature minimization using the techniques presented in section 4.2, namely the two-phase algorithm and the power density heuristic. The aim is to assess their individual and combined impact. Comparison to the state of the art is not within the scope of the experiments since previous works do not provide sufficient information to recreate a thermal simulation under the same conditions [ZML+07, CWZ04, XSY10].

Again, the same four layers configuration is used, but with additional data about the power consumption of the blocks and the thermal properties of the stack. The benchmarks are annotated with power values available at the UCLA CAD Lab website [UCL]. The power density of each block ranges...
Figure 5.8: Dimensions of the four layer stack [WFT04]

Figure 5.8: Dimensions of the four layer stack [WFT04]

from 10 to 100W/cm$^2$. The physical dimensions of the stack and its material properties are taken from the publication of Wilkerson et al. [WFT04]. This stack is depicted in Figure 5.8. The heat sink is considered to be attached to the bottom-most layer and is modeled using the default options of Hotspot (natural convection heat sink, ambient temperature of 300K).

In order to compare the impact of each proposed approach, first a reference set of values is created considering only area and wirelength minimization. To stress the temperature effect, the floorplanner has been allowed to run for more iterations than in the previous experiments, reaching more compact layouts. The resulting baseline values are reported in Table 5.5. From this table, one can notice that the maximal gradients for ami33 and, more significantly ami49, are extremely high. This can be explained by the fact that, for these benchmarks with few blocks, it is difficult to have a solution over four layers with low whitespace. In addition to this, the packing methods used in most of the floorplanners always compact the design towards one corner (lower left), accumulating the whitespace on the opposite corner (upper right) of each layer. Despite the fact that such high gradients are unac-
ceptable in real designs, these baseline values still serve for the comparative purpose of this experiment.

Three test cases are considered to verify the impact of the thermal-aware strategies proposed:

1. power density heuristic applied to area-wirelength minimization (i.e. no thermal simulations)
2. two phase algorithm
3. two phase algorithm combined with the heuristic

The grid sizes for temperature evaluation were defined according to the size of the blocks in the benchmarks\(^7\) (Table 5.6). During optimization, when only the vertical heat path is considered, a larger grid size encompassing several blocks is considered as a means to account for some of the lateral dissipation.

The comparative data is summarized in Table 5.7 and corresponds to the average of 20 runs. Case (1) shows that the simple use of the heuristic can considerably reduce temperature, even though temperature is not among the objectives of the cost function. In case (2), the explicit goal of temperature reduction does improve this metric but also forces the solution to occupy a larger area. Finally, as expected, case (3) shows better results than case (2) with further reduction not only in temperature but also in area. The shorter runtime of case (3) with respect to case (2) is an effect of the use of the heuristic. As verified in case (1) the heuristic disturbs the area minimization, therefore, in case (3), it delays the transition to the second phase, reducing the number of thermal simulations that are executed.

Although a direct comparison with previous work can not be done, the factors of temperature reduction obtained in this work are of the same order of magnitude (about 0.65x) [CWZ04, ZML+07, XSXY10]. However, in this work, area and wirelength are much lower, both before and after thermal-aware floorplanning.

Moreover, the proposed two-phase algorithm with a smooth transition between phases seems to be better than the approach used by Xiao et al. [XSXY10], where the annealer temperature is reset to a high value in the

\(^7\) The multigrid solver in Hotspot requires the grid to be a power of 2.
Table 5.5: Reference values from area-wirelength minimization

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Power (mW)</th>
<th>Area (WS%</th>
<th>Wire (µm)</th>
<th>Temp (°C)</th>
<th>Grad (°C)</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami33</td>
<td>703.5</td>
<td>355324 (22.9%)</td>
<td>24687</td>
<td>243.32</td>
<td>11.16</td>
<td>5</td>
</tr>
<tr>
<td>ami49</td>
<td>14910</td>
<td>10.48·10^6 (18.3%)</td>
<td>386205</td>
<td>221.89</td>
<td>71.35</td>
<td>13</td>
</tr>
<tr>
<td>n100</td>
<td>78.25</td>
<td>49617 (10.6%)</td>
<td>83645</td>
<td>197.02</td>
<td>1.04</td>
<td>53</td>
</tr>
<tr>
<td>n200</td>
<td>78.4</td>
<td>47857 (8.9%)</td>
<td>158640</td>
<td>206.08</td>
<td>0.72</td>
<td>279</td>
</tr>
<tr>
<td>n300</td>
<td>130.5</td>
<td>74160 (8.6%)</td>
<td>229144</td>
<td>216.63</td>
<td>1.14</td>
<td>670</td>
</tr>
</tbody>
</table>

Table 5.6: Grid size for thermal simulation

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Grid size (rows x columns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ami33</td>
<td>4x4 32x32</td>
</tr>
<tr>
<td>ami49</td>
<td>4x4 32x32</td>
</tr>
<tr>
<td>n100</td>
<td>8x8 32x32</td>
</tr>
<tr>
<td>n200</td>
<td>8x8 32x32</td>
</tr>
<tr>
<td>n300</td>
<td>16x16 64x64</td>
</tr>
</tbody>
</table>

Table 5.7: Impact of the proposed techniques on temperature and gradient reduction. Results are relative to Table 5.5.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Area-Wirelength with heuristic</th>
<th>Two-phase algorithm</th>
<th>Two-phase algorithm with heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area (WS%)</td>
<td>Wire</td>
<td>Temp</td>
</tr>
<tr>
<td>ami33</td>
<td>1.01 (28)</td>
<td>1.01</td>
<td>0.22</td>
</tr>
<tr>
<td>ami49</td>
<td>1.03 (18)</td>
<td>1.03</td>
<td>0.73</td>
</tr>
<tr>
<td>n100</td>
<td>1.01 (10)</td>
<td>1.06</td>
<td>0.77</td>
</tr>
<tr>
<td>n200</td>
<td>1.01 (8.9)</td>
<td>1.09</td>
<td>0.76</td>
</tr>
<tr>
<td>n300</td>
<td>1.01 (8.6)</td>
<td>1.08</td>
<td>0.75</td>
</tr>
<tr>
<td>Average</td>
<td>1.02</td>
<td>1.05</td>
<td>0.75</td>
</tr>
</tbody>
</table>
second phase. Even though the focus of that work is in achieving a given aspect ratio, the results reported for \( n_{100} \) and \( n_{200} \) with aspect ratios between 1.0 and 1.5 (the most common solution for these circuits), have wirelength values that are almost doubled.

In summary, the results show that the proposed heuristic performs well and is a good alternative for partitioning tools that do not interface with a thermal simulator. When a thermal simulator is used, a two-phase algorithm is an attractive option regarding runtime, as it avoids to performing the expensive temperature computation at early iterations when the floorplan is not yet sufficiently compact.

5.4 Network-aware floorplanning

The tests on the MoNICA workflow are performed based on synthetic data for the MPSoC and the traffic patterns. The objective of the tests is to compare the nested optimization approach accounting for latency, area and temperature simultaneously, with other schemes that consider these objectives separately.

For this study we considered a MPSoC having 64 cores and mapped to a 4x4x4 3D NoC. The shapes and power of the cores were generated randomly. Power densities vary from 10 to 150W/cm\(^2\), while area values vary from \( 10^4 \) to \( 5 \cdot 10^4 \mu m^2 \) with aspect ratios between 1 and 4. The power and area of the routers and networks interfaces are comprised within the cores. The stack used for thermal simulation is identical to the one used in the experiment for thermal-aware floorplanning.

CEA-Leti and UFRGS provided us with a routing algorithm, as well as with a SystemC model for 3D NoC-based MPSoCs, which is used for a fine grain simulation of the network latency. The model simulates the buffers, arbiters and protocol mechanisms for each router in the network. The routers have 7 ports: 4 for planar communication, 2 for vertical connection (up and down) and one dedicated to the core. For this experiment, the cores' operating frequency is set to 400MHz. Cores exchange messages as packets composed of 8 flits (flow control digits), with each flit having 4 bytes. The data injection rate is of 0.5 flits/cycle/core and the simulation terminates once every core
Figure 5.9: To stress the tool, the optimization start from an unfavorable configuration with respect to communication. Based on the initial mapping, the traffic pattern is set to correspond to one of two patterns: butterfly (a) and bit-complement (b). The layout of the initial mapping is shown on the left and the traffic pattern on the right.

has exchanged 80 packets.

The previous experiments in this chapter have already validated the ability of mofp to find compact floorplans and manage the thermal aspect. Here, we want to stress the new feature, i.e. the optimization of communication latency through modifications to the core-to-node mapping. Therefore, we set up the algorithm to always start from a configuration that is unfavorable with respect to latency. This unfavorable configuration is created by generating a random initial mapping and then applying a synthetic traffic pattern (bit-complement or butterfly [DYN02]). Figure 5.9 shows these initial configurations, with cores in one corner of the chip communicating with cores in the other corner. We expect the optimization process of MonICA to manipulate the mapping towards a less congested solution.

To quantify the benefits we run four test cases, T1-T4, for both traffic patterns. Each case considers a different set of objectives and allows to understand the design trade-offs. The results are listed in Table 5.8.

In T1, only latency is considered to partition the chip. The returned solution, fixing the mapping, is then optimized for area. This sequential approach is similar to those that first partition the application based on min-cut algorithms and then optimize the placement [SMBDM10, dPA10]. This gives the lowest latency, but the partitioned cores cannot be arranged compactly. The lower temperature in this case is a consequence of the large area, which
reduces the power density of the chip.

Tests T2 and T3 give lower bounds for area and temperature, allowing for a better understanding of the trade-offs in the problem. The result for T2 gives the lowest area and the highest temperature. In turn, T3 shows that temperature can be reduced at little area increase when these two objectives are considered simultaneously. In both cases, latency values are much larger than those in T1. The larger latency for T3 with respect to T2 is evidence that, for the considered MPSoC and traffic patterns, the minimization of latency and temperature are conflicting objectives.

Finally, T4 applies the MonICA workflow. With the simultaneous optimization of communication and layout, a trade-off solution can be found. At the cost of a small increase in area with respect to T2-T3 (+3%), latency is dramatically reduced to values comparable to the best solution for communication (T1).

Figure 5.10 complements the table and show results of the experiment for the butterfly traffic. T1 organizes the traffic up to the point where no vertical communication is needed, but fails on area and temperature reduction (Figure 5.10.a), while T3 fails to find a mapping providing clean routing paths (Figure 5.10.b). These deficiencies are solved in MonICA, which makes better usage of the third dimension (Figure 5.10.c).

A rigorous validation of the tool would require it to be applied to real-world designs. This was not possible in the course of this work. Nevertheless, the results presented here show that the proposed optimization strategy with a nested optimization is feasible. Moreover, we provide a demonstration of how mofp can be extended beyond the block placement problem in order to treat more specific design problems in 3D.
CHAPTER 5. EXPERIMENTAL RESULTS

Figure 5.10: Sample results for the different cases. In each figure, from left to right we have: 1) layout of the chip, 2) the communication between cores and 3) the thermal gradients in each layer. The scale of the thermal gradients vary across the plots. The maximum gradient for each case is: 8°C (a), 3.5°C (b) and 6°C. These plots are part of the morph user interface and are animated to show the progression of the optimization.

Table 5.8: NoC-aware floorplanning: four test cases and two traffic patterns (averaged over 10 runs).

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Latency (ns)</th>
<th>Area (mm²)</th>
<th>Temp (°C)</th>
<th>Runtime (s)</th>
<th>Latency (ns)</th>
<th>Area (mm²)</th>
<th>Temp (°C)</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>155.5</td>
<td>0.811</td>
<td>118.9</td>
<td>6</td>
<td>237.5</td>
<td>0.803</td>
<td>121.3</td>
<td>3</td>
</tr>
<tr>
<td>T2</td>
<td>493.6</td>
<td>0.673</td>
<td>136.7</td>
<td>114</td>
<td>485.5</td>
<td>0.673</td>
<td>136.7</td>
<td>114</td>
</tr>
<tr>
<td>T3</td>
<td>510.2</td>
<td>0.687</td>
<td>131.1</td>
<td>114</td>
<td>797.1</td>
<td>0.685</td>
<td>131.1</td>
<td>115</td>
</tr>
<tr>
<td>T4</td>
<td>196.6</td>
<td>0.708</td>
<td>129.2</td>
<td>110</td>
<td>241.2</td>
<td>0.711</td>
<td>128.4</td>
<td>105</td>
</tr>
</tbody>
</table>
6 CONCLUSION

6.1 Summary

With another dimension to explore, the 3D-IC technology greatly increases the complexity of design. The number of scenarios grows exponentially as we consider multiple dies, their ordering in the stack and the possibility of heterogeneous integration. To be able to prune this vast design space, designers must be supported by efficient automation tools from the very early stages of the design process.

This first part of the thesis has been devoted to problems in the early partitioning of 3D-ICs. Since the central point of 3D technology is the reduction of form factor, we have focused our research on the chip floorplanning problem. Besides area and wirelength, many other aspects may be important to decide the optimal partitioning, such as temperature distribution, mechanical stress or power-delivery.

To account for these manifold concerns, we opted to develop an extensible optimization framework to solve these problems. The framework, named mofp, provides a flexible implementation of the Simulated Annealing heuristic, the de facto standard for floorplan optimization [KGV83], and provides APIs for other modules to extend its functionality. These additional modules provide simulation engines to evaluate different aspects of 3D design and define specialized optimization strategies for a given type of problem.

In the course of this work we have built three modules: (i) a basic one for area-wirelength minimization, where we account for the dimensions of the TSVs during the optimization process, (ii) one integrating a thermal simulator and providing a sequential strategy for thermal-aware floorplanning and (iii) one dedicated to systems that communicate through a NoC paradigm. Statistical experiments on commonly adopted benchmarks provided evidence on the effectiveness of the proposed methods.

In addition, we studied the problem of parameterizing the SA heuristic.
This empirical process was automated with the use of another optimization algorithm, a GA, which uses a few reference circuits to learn about the impact of the different parameters of the tool on the quality of the resulting floorplan. This meta-optimization process took, in the study we have carried out, 40 hours to return a set of optimal configurations of the tool. At first sight, this two-days runtime may seem excessive and limit the interest of such approach. However, one has to put it in the perspective of the time saved by avoiding to engage a skilled human in a tedious trial-and-error process.

6.2 Perspectives

The large spectrum of contents covered in our study allows us to list several points that deserve attention in future work. We group these points in two axes of research.

Automatic parameter tuning

1. The deployment of the meta-optimization approach in a larger cluster (in this work a quad-core processor has been used) to cut runtime down to levels that allow it to be applied to more complex problems or to handle a larger set of benchmarks in the training phase;

2. The use of reference points for Pareto optimization as a means to ignore the extremities of the front and focus the search at trade-off solutions (in the center of the Pareto curve). In this work, this is partially implemented as we normalize the objectives of the meta-optimization by results taken from the literature on floorplanning. However, an improvement should be perceived if we also apply changes to the fitness assignment method (e.g. crowding distance) used in the GA’s selection mechanism [DSNC06];

3. The more aggressive approach of letting the SA process itself tune the parameters during its execution. To this point, game theory [Cam11] can be a starting point to build up a policy to dynamically increment/decrement the probability of the heuristic moves according their success ratio.
6.2. PERSPECTIVES

Figure 6.1: Two analysis of the same layout (c) with the original hotspot routines and (d) overlaying the positions and thermal properties of the TSVs. All five blocks have the same power density (10W/cm²)

Development of new features in mofp

1. Integrate the impact of TSVs on the thermal profile of the chip. The decrease in TSVs dimensions, the use of hollow TSVs (i.e. less metal) and the increasing thermal conductivity of bonding materials may change the present idea that TSVs can be used to propagate the heat out of the chip. To investigate this, we are currently validating modifications to the hotspot simulation engine to account for a mix of material with different thermal conductivities (Figure 6.1);

2. Continue the development of MoNICA, working with CEA-Leti/UFRGS to apply it to an industrial test case. Other axes of research are (i) to consider partially connected topologies [DSPBed] and to co-optimize the position of the vertical links and (ii) to perform the optimization of a system for more than one traffic pattern in order to account for the reconfigurability of the MPSoC [vSP12];

3. Consider stacks with dies from different technologies and account for their differences in terms of economical cost/yield and performance. Scaling
factors on area, power and performance can be applied when moving blocks between layers from different technology nodes during the optimization process. The designer can, in this way, leverage the possibility of combining aggressive and mature technologies to design ultra-low-power applications with improved performance. For digital blocks the scaling trends are well known and easy to implement. However, to deal with blocks from domains that do not follow the same scaling trends (e.g. analog, optics), we need to make use of knowledge databases, collecting the characteristics of previously designed systems to predict their behavior under different design conditions.
PART II

HIERARCHICAL DESIGN OF HETEROGENEOUS SYSTEMS

7. Introduction
   7.1. Basic concepts in automated design
   7.2. A review of hierarchical design methods
   7.3. Discussion and motivation

8. Predictive hierarchical synthesis using Pareto-fronts
   8.1. Design flow
      8.1.1. Characterization of Pareto-fronts
      8.1.2. Predictive model
      8.1.3. Constraining high-level optimization
      8.1.4. Multiple levels of abstraction and hierarchical synthesis
      8.1.5. Summary
   8.2. Pareto-front computation by numerical optimization
      8.2.1. Normal Boundary Intersection method
      8.2.2. Methods for solving the constrained single-objective problems
   8.3. Building predictive models
      8.3.1. Basic structure of the predictive model
      8.3.2. Change of variables
      8.3.3. Identification of admissible parameterizations for \( G(F^0) \)
      8.3.4. Interpolation with radial basis functions
      8.3.5. Approximation of the validity domain \( \Omega \)
   8.4. Accounting for environment parameters
   8.5. Summary
9. Experimental results 122
   9.1. Implementation and experimental method .......................... 122
      9.1.1. Implementation of the framework ............................ 122
      9.1.2. Methodology .................................................. 123
   9.2. Data link for optical NoC application ............................. 124
      9.2.1. Description ................................................... 124
      9.2.2. Results ....................................................... 125
   9.3. Operational Transconductance Amplifier (OTA) ..................... 130
      9.3.1. Description ................................................... 130
      9.3.2. Results ....................................................... 131
   9.4. Non-monotonic Pareto Front ......................................... 133
      9.4.1. Description ................................................... 133
      9.4.2. Results ....................................................... 134
10. Conclusion 138
    10.1. Summary ....................................................... 138
    10.2. Perspectives ................................................... 139
The current advances in packaging technologies enable the tight integration of functions from different physical domains (e.g. mechanical, optical, biological) in a single chip. This new generation of heterogeneous systems unveils unprecedented possibilities for innovation in key market segments such as health care, transportation and security [ZR09]. However, the stringent requirements that accompany miniaturization (e.g. low power, cost efficiency) demands an increasing effort in the design phase. According to the International Technology Roadmap for Semiconductors [ITR11], an early bottleneck slowing down the adoption of emerging technologies is the lack of design tools that can manage heterogeneous content throughout the V-design cycle of the system (Figure 7.1).

The design of heterogeneous systems is marked by ad-hoc design flows that combine different simulators and models (Figure 7.2). To manage these disjoint flows, hierarchical decomposition is a fundamental concept. It allows to separate design concerns and divide the problem into simpler, tractable parts. In one hand, by dividing a system into smaller blocks based on their domains (e.g. electronics, optics, fluidics), each block can be modeled and

![Figure 7.1: V-cycle for system design (adapted from [ITR11])](image-url)
designed using specialized CAD tools. On the other hand, by abstracting the low-level details of the blocks, it is possible to describe the global functionality of the system in a single model. The description of these high-level models is done using behavioral languages such as SystemC-AMS, VHDL-AMS [PLV05] or Simulink [Mat]. These languages allow to combine different Models of Computation (MoC) and thus provide means to describe phenomena from different domains in a single environment [EBGV09].

The hierarchy of models constructed on the basis of structural decomposition and abstraction enables a designer to explore the design space at different levels. System-wide trade-offs can be assessed using a high-level model. Architectural decisions taken at this level are then propagated to the next lower levels. While hierarchical design and automatic synthesis tools are well established in the domain of digital circuits [DM94], the same is not true for AMS systems. Contrary to digital, where the binary decision levels make consecutive blocks loosely couple, in AMS design sub-systems interact with each other in a stronger way. The aim of this part of the thesis is to provide systematic (and therefore automatable) means to explore the performance trade-off of individual blocks to achieve system-level optimality and promote design reuse.

7.1 Basic concepts in automated design

The goal of a design process is to size every block of a system so that the performance requirements at the system level are fulfilled. This section introduces the concepts and notations used to express the design problem in
such a manner that it can be treated in an automated way. Even though a design process may involve several abstraction levels, it is sufficient to state the definitions only in terms of two levels. This is because at any one point in the design process the problem can be formulated in terms of performance requirements and design parameters. The terms used here are block and system to denote lower and higher abstraction levels respectively. The main definitions are given using the block, and the system is introduced when explaining the transitions between hierarchical levels.

![Diagram of simulation model](image)

**Figure 7.3**: A simulation model

**Simulation model**

The starting point for an automated design process is to establish a parameterizable simulation model of the block under design. A simulation model is composed not only of the model of the block but also of a testbench (Figure 7.3). The model of the block describes its intrinsic structure which is subject to sizing via design parameters, while the testbench models the surrounding environment. The testbench defines loading conditions and excitation signals that allow the performance of the block to be measured in a realistic context. Additionally, the testbench can define a startup sequence for the simulation software, loading and configuring technology databases used in simulation. Having a parameterizable testbench is key to enable the reuse of a simulation model and the construction of a library of components. From now on, this indissociable combination of the block model and its testbench will be referred to simply as a model and treated as a function \( F(\cdot) \) that maps design and environment parameters to a set of performance metrics.
The admissible values for the design parameters constitute the parameter space $\mathcal{X}$, representing all possible realizations of a block. By symmetry, environment parameters also constitute an environment space $\mathcal{E}$. However, as these remain fixed during the design process, we may omit them from the discussion. The image of the parameter space for a given environment is the performance space $\mathcal{F}$ (Figure 7.4). In this work, the term design space is used as a general reference to all $\mathcal{X}$, $\mathcal{E}$, and $\mathcal{F}$, i.e. all the spaces that are necessary to determine the characteristics of a design.

![Figure 7.4: The parameter space $\mathcal{X}$ and the performance space $\mathcal{F}$.

Sizing constraints

In the definition of the parameter space, the term admissible values implies that some conditions have to be satisfied. These conditions are called sizing constraints and can be divided into dimensional and functional types. Dimensional constraints are those that can be directly checked from the values of the parameters $\mathbf{x}$, thus explicitly limiting the parameter space. The most general are bound constraints giving lower and upper limits to the parameters according to what is feasible in a given technology process. More specific conditions, for example, would be to require one parameter to always be kept superior to another ($x_1 \geq x_2$), or to have the sum of two parameters adding up to a constant ($x_1 + x_2 = a$).

Functional constraints express more intrinsic conditions necessary for the
7.1. BASIC CONCEPTS IN AUTOMATED DESIGN

Correct operation of the block (such as, for example, to constrain all transistors to the saturation region in an amplifier circuit). The verification of functional constraints is not as direct as that of dimensional constraints. It typically involves performing a partial simulation of the model. If the functional constraints are satisfied, then the simulation can be completed. However, if a violation is detected, performance evaluation becomes meaningless and the simulation can be terminated.

Performance requirements

While sizing constraints impose restrictions on parameter values, requirements impose conditions on the performance metrics. Performance requirements are indeed the focus of the design process. Requirements state what a block must achieve, regardless of the implementation of the block. Thus, even though the sizing constraints are unique for each possible architecture of a block, the formulation of the performance requirements remains unchanged. Certainly, the performance that can be achieved depends on the architecture of the block.

Requirements are divided into two categories: objectives and constraints. Constraints establish the minimum criteria to accept a design and should be strictly satisfied (e.g. minimum throughput, maximum power). Objectives are less restrictive. They do not define values, but directions for improvement. Objectives are stated as maximization or minimization (e.g. maximize gain, minimize area). Thus, when all constraints are met, a design solution that improves the objective is still preferred. When no further improvement in the objective is possible, the design is said to be optimal.

Automatic design

An automatic design process is capable of finding a set of design parameters that fulfill the requirements under given environment conditions. The elements involved in the flow are illustrated in Figure 7.5.

Two kinds of automatic design exist: optimization-based and rule-based. In rule-based methods, designers translate a manual approach into an automatic process. Just like a designer, these methods process the problem
sequentially, applying an analytical procedure to fix some design parameters before processing the others [DRPCFB06]. There is little or no need for simulation during the execution of the procedure, which makes these methods extremely fast to execute. However, the rules are specific for a given architecture of a block and the time spent establishing them is non-negligible. Additionally, the sequential application of the rules imposes a well defined route through the design space, which guarantees feasibility, but may exclude the ability of finding optimal solutions.

Optimization-based methods are completely different, requiring no deep expertise about the block under design but demanding a much higher computational effort. A numerical optimization algorithm seeks to satisfy the requirements by modifying the design parameters iteratively. New design points are compared to the previous ones to determine new search directions. The search continues until it converges to a solution where no further improvements can be made, i.e. to an optimal solution. By determining search directions on a quantitative basis, optimization algorithms can apply parameter variations which no designer would consider and explore the totality of the design space.

Of course, optimization- and rule-based methods can be complementary techniques, balancing fast convergence with good exploratory capabilities [OK00]. Nevertheless, the focus of this research is on general design methodologies. Therefore the following discussion considers only automatic design via numerical optimization.
7.1. BASIC CONCEPTS IN AUTOMATED DESIGN

Figure 7.6: The parameter space $\mathcal{X}$ and the performance space $\mathcal{F}$.  

Optimization, optimality and trade-offs  

The principles of optimization methods were presented in sections 3.1.1 and 3.2, but we shall rediscuss some notions to place them in the context of the constrained optimization of physical systems.

Every practical design problem involves the simultaneous optimization of multiple competing performance metrics and, consequently, leads to trade-off situations. Using the notation introduced in this chapter, the MO problem is written as:

$$
\min_{x \in \mathcal{X}} \mathbf{f} = F(x) = \begin{bmatrix}
f_1(x) \\
\vdots \\
f_n(x)
\end{bmatrix} \quad \text{subject to: } \begin{cases}
c_{eq}(x) = 0 \\
c_{ineq}(x) \leq 0
\end{cases} \quad (7.2)
$$

where $c_{eq}, c_{ineq}$ are all sizing and functional constraints formulated as equalities and inequalities.

The solution of (7.2) leads to Pareto-optimal designs, where improving one performance can only be done at the cost of another. The set of all these optimal designs is the so called Pareto-frontier. The term Pareto-front (PF) is typically used to refer to points in the performance space, whereas their counterparts in the parameter space are called the Pareto-set (PS). The two sets are linked via the model function $F(\cdot)$ as illustrated in Figure 7.6.

\footnote{By convention, minimization of objectives is considered for optimization. The maximization of an objective is obtained by multiplying it by -1.}
Relationship between hierarchical levels

Hierarchical design is characterized by a direct correspondence between the performance metrics of a block and the design parameters of the system. Consider the example in Figure 7.7. At system level, the designer of a control system manipulates the poles and zeros of the controllers $K_1$ and $K_2$ in order to achieve a smooth regulation of the plant $A$. Considering that the controllers are implemented as analog circuits, the problem at the block level is to manipulate the widths ($w$) and lengths ($l$) of transistors to obtain the required poles and zeros. In this example one can state the mapping functions $F(\cdot)$ for each component as:

$$F_{K_1}: w, l \rightarrow K_1\text{poles, zeros}$$

$$F_{K_2}: w, l \rightarrow K_2\text{poles, zeros}$$

$$F_{\text{system}}: K_1\text{poles, zeros}, K_2\text{poles, zeros} \rightarrow \text{system settling, damping}$$

From these expressions appears the equivalence between the performance metrics of the blocks and the design parameters of the system they compose, i.e.:

$$x_{\text{blocks}} \rightarrow f_{\text{blocks}} \equiv x_{\text{system}} \rightarrow f_{\text{system}}$$

This relation allows navigation between levels. In bottom-up transitions, from blocks to system, the performance of the blocks allow the direct evaluation of the system performance. Similarly, in a top-down transition the system design parameters become performance requirements that the blocks must satisfy.
7.2. A REVIEW OF HIERARCHICAL DESIGN METHODS

must satisfy.

The main difficulty in hierarchical design lies in the top-down transitions. An optimal set of design parameters for the system may ultimately result in a too ambitious requirement for the blocks and, thus, be unfeasible. Such unsolvable situations require a redesign of the system and should, therefore, be avoided. The next section gives a review of the main methodologies to manage the design of hierarchical systems.

7.2 A review of hierarchical design methods

The use of high-level languages and hierarchical modeling is necessary to handle the complexity of systems that span multiple physical domains. As a consequence, the engineer has to manage the joint design of small sub-systems which, when assembled together, meet the required performance levels. To start this review, we recall the two main paradigms of hierarchical design:

**Bottom-up design (BU):** This paradigm trades optimality for component reusability. Complex systems are created by assembling elementary blocks that are already designed and available in the form of a component library. As a result, the performance level of the system varies discretely according to the choices of which elementary blocks are combined. With this coarse control of the result, it is clear that the design will be either underdesigned or overdesigned with respect to the requirements. Nevertheless, this is a very attractive approach in terms of productivity and can become efficient if the libraries provide a very large choice of components (e.g. the case of digital design with standard cells with well-controlled functionality).

**Top-down (TD) constraint propagation:** Given the requirements on the system, the engineer first solves the optimal design problem using a system-level model. The system-level design parameters are then derived into requirements for the blocks and the design process continues down the hierarchy [CCC+96]. However, this top-down propagation of requirements is not obvious because there may be multiple ways in which to split a requirement over the sub-systems (e.g. the gain of each stage of an amplifier). The derived requirements may either not exploit the best capabilities
CHAPTER 7. INTRODUCTION

of sub-blocks (leading to oversized designs), or be too ambitious and cause the design process to fail. As a consequence, this task must be performed by an expert designer, who can foresee which blocks are harder to design and also infer trade-off relations that were not captured by the system-level model.

These two opposite paradigms show how intricate the design of hierarchical systems is: with requirements flowing from the top-down, and feasibility limitations being imposed from the bottom-up (Figure 7.8). To handle this problem, some authors have presented general purpose environments capable of managing hierarchies and optimization cycles but leaving to the user the definition of a constraint propagation strategy [NM10]. Others, have focused on very specific applications and used domain knowledge to build fully automatic flows [MPSG11]. However, these frameworks rely heavily on expert knowledge to define design rules, inheriting all the drawbacks that this implies (i.e. hard to develop, maintain or use).

In response to this situation, several research groups worked on a more general approach that is based not on design rules, but rather on the explicit characterization of the feasibility region of lower-level blocks and their use to constrain a TD flow, guaranteeing that any requirement propagated top-down is feasible. Figure 7.9 illustrates this hierarchical design process comprising a bottom-up characterization step and the top-down sizing of the

![Figure 7.8: Relation between hierarchical levels in the design process.](image_url)
7.2. A REVIEW OF HIERARCHICAL DESIGN METHODS

Figure 7.9: The five steps in state of the art hierarchical design (adapted from [MG09]).

...system. The systematic incorporation of low-level information in the TD design process provides a means to truly automate the synthesis and also fosters the reuse of already characterized blocks in different applications. Among the different implementations of this flow we can distinguish two distinct approaches for the bottom-up characterization:

**Modeling of the entire feasible region**: A series of (random) simulations are run to obtain an image of the performance space \( \mathcal{F} \) of a low-level block. The boundary of the space \( \mathcal{F} \), limiting what is feasible and what is not, is approximated using polytopes [SGA07] or a non-linear classification method, such as Support Vector Machines (SVM). A TD design is conducted under the constraint that the solution at each level lies inside the feasible region of the next lower-level blocks. Among the many demonstrations of this method [RH96, DBNSV05, DBNV06, NSW+10], some also consider changing environment parameters (e.g. output impedance, drive current) when characterizing the feasible region [Sun11]. This brings more power to the TD flow, because it allows to express how the interaction between two connected blocks impacts their performance, an impact that is difficult to capture at the system-level model, where low-level details are omitted.

**Modeling the Pareto Front (PF)**: To model the boundary of the feasible region, a large number of designs has to be simulated. Many of these, although feasible, are far from being optimal and do not present any interest...
for design. Thus, researchers have proposed to model only the part of the boundary corresponding to optimal designs: the PF. The principle is called Multiobjective Bottom-up (MUBU) [EMG05] and is illustrated in Figure 7.10. The PF of low-level blocks is characterized by a set of discrete points obtained through multiobjective optimization, preferably employing some strategy to obtain an evenly distributed coverage of the PF [SGA07]. These discrete points are propagated up the hierarchy and combined to compute the Pareto-front of the higher-level block. This is exactly the BU approach described in the beginning of this section. However, since each block is characterized by multiple points spanning the entire trade-off curve, the MUBU method can offer the designer a large number of choices. Another important aspect of designing based on PFs is that several architectures of a system can be compared in terms of their PF. The designer can thus superpose trade-off curves and clearly identify under which conditions one architecture performs better than another [ESG07, RCLF09].

While the initial proposal of MUBU was stated as a BU only approach, the characterized PF can also serve to constrain a TD flow. The principle is to use the optimal trade-off curves of the low-level blocks to calibrate the behavior of the system-level model. Figure 7.11 illustrates well this approach in the design of a Phase-Locked Loop (PLL) circuit [RGR07]. In this example, the PF of a block at transistor level is modeled by a polynomial and embedded in a Verilog-AMS model used at system-level. As a consequence, the TD flow optimizes the design parameters of high-level models.
7.3. DISCUSSION AND MOTIVATION

that behave as optimal designs and the requirements propagated down the hierarchy are not only feasible, but also optimal with respect to the system requirements. Several other examples demonstrate the power of this approach [ZMGS06, GEMM07], which is considered the state of the art of hierarchical synthesis and has been extended to support robust design via yield-aware Pareto-fronts [TTR06, MGG10].

7.3 Discussion and motivation

Heterogeneous systems are diverse in nature and applications. Contrary to analog design, where workhorse circuits like the OpAmp make the development of specialized design flows viable, the research on heterogeneous design has to focus on general purpose design techniques. To this point, hierarchical design with PFs seems the most promising option and serves as the basis for this work.

The original MUBU approach based on the sole combination of discrete design points allows a fast evaluation of high-level trade-offs. However, it provides no control on how representative the higher-level PF is. In the
Figure 7.12: The Pareto Front of system A can be obtained by (a) combining the individual solutions found for the lower-level blocks or (b) interpolating the lower-level curves to better satisfy the system-level constraints. The dotted lines mark the regions of the lower-level fronts that actually contribute to the feasible high-level front.

presence of non-linearities, the combination of two low-level PFs with evenly distributed points will rarely produce an even coverage of the high-level PF. Moreover, strong functional requirements at the system-level (e.g. a minimum efficiency to satisfy regulation standards) can restrain the exploitable region of the system PF, as illustrated in Figure 7.12a. Such limitations disappear when a continuous representation of the low-level PF is used at the high-level optimization, because the optimization algorithm will be free to explore the portions of the low-level PFs that really contribute to the system PF (Figure 7.12b).

However, the flows proposed so far have only exploited continuous representations of the PF and ignored the role that its counterpart in the parameter space, the PS, can play in the top-down sizing. The flow depicted in Figure 7.9 requires an optimization to be run at every level. Nevertheless, since the system model embeds the block’s PF, the requirement propagated down corresponds, necessarily, to a point in the PF of the block. Consequently, the solution for the block design belongs to its PS. Thus, one can use interpolation methods between the PF and the PS to immediately retrieve the desired solution and avoid costly optimization loops.
The present work is motivated by this opportunity to accelerate the design cycle by means of simple data mining and capitalize on the computational effort spent during the bottom-up characterization process. In the next chapter we present a general purpose design framework where both PF and PS of a block are encapsulated in what we call a \textit{predictive model}. A predictive model establishes a continuous mapping from points in the PF to the PS, allowing the design parameters to be retrieved immediately for any possible performance trade-off (Figure 7.13). The main contributions of this work are:

- The formalization of the complete design flow.
- The definition of the \textit{predictive model}, describing its interface and proposing a concrete implementation that is self-contained, reusable and language-agnostic;
- The composition of low-level predictive models with a high-level simulation model without incurring any modification to the simulation model;
- The use of this composition in a conventional optimization loop, demonstrating the natural integration of the predictive models in the design flow;
- The inclusion of environment parameters in the characterization of the \textit{predictive models} to increase design reuse.
The success of top-down synthesis depends on the ability to foresee the trade-offs that a given low-level implementation imposes at the scale of the system. To avoid unfeasible designs, state of the art methods characterize the capabilities of elementary blocks first and use this information to constrain the system-level specification. The characterization steps at low abstraction levels are computationally expensive. Consequently, the collected data must be efficiently exploited to accelerate the subsequent synthesis steps. This chapter presents a synthesis flow where data interpolation is used to quickly predict the optimal solution of a design problem without incurring the need for additional optimization runs.

8.1 Design flow

In the first section of this chapter we will discuss the overall flow, allowing the reader to have a clear understanding of the working of the framework. The elements in the flow that require a deeper understanding of optimization and interpolation techniques are then detailed in separate sections.

A complete view of the framework is illustrated in Figure 8.1. In the center, appear the three main processes (1–2–3) necessary to construct the predictive models of elementary blocks and associate them with higher-level models to build hierarchies. These processes are supported by optimization and interpolation methods, while interfaces to external simulation software define a clear separation between the synthesis and the simulation environments. The predictive synthesis process (4), on the right, is performed entirely within the framework and allows to go from the performance requirements to the sizing of the complete system. The resulting sizing is then subjected to a traditional BU verification step (5) executing the simulation models composing the system.
Figure 8.1: Global view of the synthesis framework
8.1. DESIGN FLOW

8.1.1 Characterization of Pareto-fronts

The computation of the optimal performance trade-offs of a block is the initial point of the flow. This task is performed by a multiobjective optimization method applied to the block simulation model (Figure 8.2) and results in a discrete number of optimal solutions \((x^*, f^*)\). The choice of the best suited optimization method is problem dependent. However, it is necessary to employ some strategy guaranteeing an homogeneous discretization of the PF, i.e. an even spread of points over the full span of the PF [DD98, ZL07, EU11].

An important consideration in this step is the computational runtime. In a bi-objective problem, the PF being discretized is a 2D curve. With three objectives it is a surface, with four it is a volume and so on. Thus the number of points necessary to represent the trade-off grows exponentially with the number of objectives considered. The designer has to use his knowledge about the architecture of the block to identify which metrics do really represent trade-offs, and by doing so, limit the dimensionality of PF.

The performance vector \(f\) returned by a simulation model can give access to multiple metrics (e.g. power, area, gain, noise, output current, etc). Yet, not all of them are competing or of major interest for the design. Thus, we define two classes of metrics:

**Functional performance metrics**: those that closely represent the function performed by the block. For example, gain and noise for a Low Noise Amplifier, conversion error and power for an ADC, bandwidth and attenuation of a communication link.

**Para-functional performance metrics**: considered as of secondary im-
portance, typically because they are correlated to one of the functional metrics. In addition, metrics that should always be minimized, independently of the functionality of the block, such as area.

It is up to the designer to decide on the classification. The multiobjective optimization will then only consider the trade-off between functional metrics to compute the PF. Para-functional metrics will be simply evaluated and stored for future prediction. Optionally, the para-functional metrics can be optimized after the computation of the PF, as a local search in the vicinity of the Pareto solutions. This post-optimization can serve, for example, to enforce solutions trading gain and bandwidth in an amplifier (functional metrics) to also present minimal area.

The performance vector \( \mathbf{f} \) can be split into these two types of metrics, i.e. \( \mathbf{f} = [f_{\text{func}}, f_{\text{para}}] \). However, only the functional metrics will play a role in the predictive models. Thus, for simplicity in the notation, we will consider \( \mathbf{f} = f_{\text{func}} \), unless otherwise stated.

### 8.1.2 Predictive model

The optimal design points \((\mathbf{x}^*, \mathbf{f}^*)\) collected during the characterization of the PF can be exploited in two ways that are useful for the design process:

1. The performance trade-off curves are used to *make decisions* at a high level, abstracting the design parameters.
2. For a selected trade-off \( \mathbf{f}^* \), tracing back the corresponding parameters \( \mathbf{x}^* \) allows to *implement* the design.
The predictive model encapsulates the design data in such a way that both these operations can be performed easily, and that the data can be seamlessly integrated into a hierarchical design flow. The encapsulation is called a predictive model because it uses interpolation techniques to reconstruct the PF and predict the characteristics of design points that were not simulated during characterization.

The concept of the predictive model is presented in Figure 8.3, where its interface is shown by the rectangle with cut corners. It takes as input a subset of the performance vector (the independent variables of the interpolation, noted \( \mathbf{f}^\circ \)) and predicts the remaining performance metrics and design parameters. Thus, the prediction function \( G(\mathbf{f}^\circ) \) combines two mappings together, each corresponding to one of the enumerated design tasks:

\[
[f, x] = G(\mathbf{f}^\circ) \equiv \begin{cases} 
  \mathbf{f} = G_f(\mathbf{f}^\circ) & \text{(performance prediction)} \\
  x = G_x(\mathbf{f}^\circ) & \text{(parameter prediction)}
\end{cases}
\]  

(8.1)

Figure 8.4 illustrates these functions. On the left we have a bi-objective PF between \( f_1 \) and \( f_2 \) represented by the solid line and the corresponding evolution of the design variable \( x_1 \). The two curves interpolating the sampled points are parameterized as functions of \( \mathbf{f}^\circ = [f_1] \). On the right we have a 3D example, with the trade-off surface parameterized by two metrics in order to predict the third one; in this case, \( \mathbf{f}^\circ = [f_1, f_2] \).

The figure also shows the domain \( \Omega \) in which the prediction is valid. This domain is the projection of the PF on the plane of the interpolation variables.
For the outputs of the function $G(\cdot)$ to be reliable, its input $f^\circ$ must be explicitly constrained to lie inside the bounds of $\Omega$. Such a constraint has to be embedded in the model, so that it can be correctly used in an automated design flow. In the two-dimensional case, this is straightforward, because one can clearly identify lower and upper bounds for the input: $f_1^\circ \leq f_1 \leq f_1^\ast$. In higher dimensions, however, upper and lower bounds defining box constraints are not sufficient. In these cases, the general solution is to use surface reconstruction methods to model the boundary $\partial \Omega$ and establish a signed distance function to test the validity of the input [Cha07, EI09]. This function has the following property:

$$v(f^\circ) = \begin{cases} 
> 0 & \text{if } f^\circ \in \Omega^c \text{ (outside)} \\
0 & \text{if } f^\circ \in \partial \Omega \text{ (on the boundary)} \\
< 0 & \text{if } f^\circ \in \Omega \text{ (inside)}
\end{cases}$$ (8.2)

Thus, if $v(\cdot)$ has a negative value, the results returned by the predictive model $G(\cdot)$ are trustworthy. Moreover, the magnitude of $v(\cdot)$ is made proportional to the distance to the boundary $\partial \Omega$. A positive value that decreases, for example, indicates that the input is getting closer to the valid domain. Therefore, any process using the predictive model can find the valid domain by following a descent direction of $v(\cdot)$. This will be the case in the next step, where predictive models will be used inside an optimization loop and $v(\cdot) \leq 0$ will be added to the inequality constraints to be satisfied. So, the interface of the predictive model is reviewed, embedding the function $v(\cdot)$ and outputting its value (Figure 8.5).

**Figure 8.5:** Interface including the output value of the validity function $v(f^\circ)$.

An important aspect of the predictive model that deserves to be stressed is that it is a pure, self-contained, data mining operation, simply mapping points in one space to another. It completely abstracts the semantics of the simulation model it represents. This language-agnostic property is key
8.1. DESIGN FLOW

to heterogeneous designs, since the predictive model can be easily shared between project teams and exploited in environments without the tool chain to simulate the original model.

8.1.3 Constraining high-level optimization

In hierarchical design, the performance metrics of lower-level blocks are considered as design variables at the higher level. For convenience we recall the relation (7.6) here:

\[ x_{\text{blocks}} \rightarrow f_{\text{blocks}} \equiv x_{\text{system}} \rightarrow f_{\text{system}} \]

Thus, low-level PFs defining limits on the values of \( f_{\text{blocks}} \) can be used to constrain the values of \( x_{\text{system}} \). In this manner, the exploration at system-level will explore the PFs of the blocks in order to find the combination of parameters that yield the best performance at the scale of the system.

In the introduction, we have indicated that previous works have implemented this constraint by incorporating a polynomial equation of the block PF into the simulation model of the system. Such an approach, while correct, goes against the principles of modularity and reuse because it specializes the high-level simulation model for a given low-level architecture.

Instead, in this work we preserve the simulation models intact and create specializations by an assemblage of predictive models. The scheme we devise to perform high-level optimization is depicted in Figure 8.6, which we detail next.

The system \( A \) is implemented by the sub-blocks \( B \) and \( C \), and therefore computes its system-level performance \( f_A \) based on the performance of the sub-blocks: \( x_A = [f_B, f_C] \). To achieve the requirements, the optimization algorithm does not directly manipulate the parameters of \( A \) but it operates on the inputs of the predictive models of \( B \) and \( C \) (\( f_B^0 \) and \( f_C^0 \)), respecting their domain of validity (\( [v_b, v_c] \leq 0 \)). This reduces the design space of \( A \) to the PFs of its blocks.

The solution of this optimization process returns the performance-parameter pair not only for the system but also for the blocks, allowing movement between high-level requirements and low-level sizing. Moreover, the designer
CHAPTER 8. PREDICTIVE HIERARCHICAL SYNTHESIS USING PARETO-FRONTS

Input: High-level requirements

<table>
<thead>
<tr>
<th>Block</th>
<th>Performance Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>$f_B$, $x_B$</td>
</tr>
<tr>
<td>C</td>
<td>$f_C$, $x_C$</td>
</tr>
</tbody>
</table>

$x_{opt} = [f_B, f_C]$ = $x_A$

Output: optimal performance-parameter pairs for high-level ($f_A$, $x_A$) and lower-levels ($f_B$, $x_B$), ($f_C$, $x_C$)

**Figure 8.6:** Scheme to optimize a system $A$ that is implemented by blocks $B$ and $C$. The simulation model $A$ is the only element evaluated outside the synthesis environment.

can easily conduct an architectural exploration by swapping the predictive model of a block with another representing a different topology/architecture implementing the same functionality.

### 8.1.4 Multiple levels of abstraction and hierarchical synthesis

The proposed composition of simulation and predictive models is transparent for the optimization method. Thanks to this, handling multiple levels of abstraction is straightforward. The same steps used to generate the predictive models of elementary blocks can be repeated to characterize complex architectures. Figure 8.8 illustrates it on the same example of a system $A$ with two blocks $B, C$. The composition is treated as whole, its trade-offs are characterized and the resulting predictive model is stored in the library for reuse. The cycle can be repeated to encapsulate systems of growing complexity.

The models created this way are tied to the choice of the blocks used in the composition. Thus the complex predictive model in the example is noted $A_{B,C}$ to mark its hierarchical relation to $B$ and $C$. If another architecture is considered, replacing $B$ by $B'$ for example, then the characterization process is executed again to give the alternative model $A_{B',C}$. 

Thesis on Micro et Nanotechnologies 92 Felipe Frantz Ferreira / 2012 Ecole Centrale Lyon
These complex hierarchical systems are built bottom-up, using the PF of the lower-levels to constrain the design space. Symmetrically, the design process is made by a top-down traversal of the predictive model hierarchy retrieving points in the PS. Figure 8.8 summarizes the flow. The prediction function \( G(\cdot) \) for the blocks at each level is split into its two mappings: \( G_f(\cdot) \) interpolating the PF, and \( G_x(\cdot) \) interpolating the PS. The reader can notice the distinct role of each one, \( G_f(\cdot) \) being necessary for the BU characterization and \( G_x(\cdot) \) for the TD synthesis.

A designer using this methodology to size the system \( A \), only has to specify a desired performance trade-off that is within the domain covered by the system predictive model. The framework will then query every sub-block in the hierarchy and return the full sizing at every level. This process is computationally inexpensive as it does not perform any iterative simulation of the models.

Nevertheless, the returned sizing is a prediction that is subject to the interpolation error of the function \( G(\cdot) \). This error should be small if the PF is discretized by a sufficient number of points. However, it is hard to establish what can be considered to be a sufficient discretization. In all cases, the designer must verify the adequacy of the predicted sizing by performing a bottom-up verification using the simulation models. If the requirements are not met, the designer can compare the predicted and effective performance for each block in the hierarchy and identify that (or those) which presents the largest prediction error. The designer can then re-optimize those blocks separately and/or run an additional characterization step to add more points to their PFs.

**Figure 8.7:** Encapsulating systems of growing complexity. The predictive model for the composition is noted \( A_{B,C} \) to mark its dependency to \( B \) and \( C \).
8.1.5 Summary

At this point we conclude the presentation of the framework from the perspective of the designer. We have given a definition for the predictive model and also shown how it is used to accelerate design and promote reuse. The computation of the PF and its interpolation have been broadly described. In the next sections, we detail these two tasks, discussing alternatives and presenting our implementation.
8.2 Pareto-front computation by numerical optimization

A reliable prediction depends entirely on a representative discretization of the PF, covering the full span of the PF with an even distribution of points. The most basic approach to discretize the PF is to convert the MO problem (7.2) into multiple SO problems where a weighted sum of the multiple objectives is minimized:

$$\min_{x} \sum_{i=1}^{n} w_i \cdot f_i(x) \quad \text{s.t.:} \begin{cases} c_{eq}(x) = 0 \\ c_{ineq}(x) \leq 0 \end{cases} \quad (8.3)$$

Different points in the PF are obtained for different convex combinations of the weights (i.e. $\sum w_i = 1$). This approach is easily understood by designers: a gradual transition from weights $\{w_1 = 1, w_2 = 0\}$ to $\{w_1 = 0, w_2 = 1\}$ corresponds to a gradual change from the minimization of $f_1$ to that of $f_2$. However, the intuitive procedure of solving (8.3) for constant increments of the weights does not necessarily lead to an even coverage of the PF (Figure 8.9a). The solutions are, in fact, concentrated in regions where the curvature of the PF is relatively high (knee points). Moreover, this formulation cannot find solutions where the PF is concave [DD97].

Another approach to obtain a discretization of the PF is to solve (7.2) using a Multiobjective Evolutionary Algorithm (MOEA – as discussed in Section 3.2 for the unconstrained case). MOEAs can solve difficult optimization problems (e.g. non-smooth, noisy), but the typical implementation of these algorithms presents some difficulties in the context of this work. Firstly, modifications to the usual selection mechanisms are necessary to avoid premature convergence in problems with more than two objectives (many-objective problems) [ITN08]. Secondly, if a portion of the performance space frontier $\partial F$ is dominated, it will be excluded from the search yielding a so-called discontinuous PF, although $\partial F$ is continuous (Figure 8.9b). Finally, there is no consensus technique to handle constraints without compromising the exploratory capabilities of the algorithms [CC12]. These last two points limit the application of typical MOEAs for PF computation in our framework, where design problems have to satisfy several sizing constraints and where a faithful sampling of $\partial F$, including points in dominated regions, is required.
for correct interpolation of the data\textsuperscript{1}.

These limitations can be circumvented by another class of methods, called Boundary Intersection (BI) methods. BI methods are similar to the weighted sum approach in that they also decompose the MO problem, but they introduce additional constraints to find solutions that intersect $\partial F$ at specific locations and can, thus, deliver an even discretization of $\partial F$ (Figure 8.9c). The idea dates back to the early 80’s [PP80, Spa82], but has gained popularity under the name of Normal Boundary Intersection (NBI) after a more recent publication [DD98]. It has, since, been used in many interesting implementations [ZL07, MGGS09, EU11]. The following section details the NBI method and highlights points that are important for the later task of interpolation. From now on, we consider that the term PF covers the ensemble of points in $\partial F$, i.e. dominated and non-dominated. This corresponds to the usage we will make of these points to build a continuous representation of the trade-off curve.

\subsection{Normal Boundary Intersection method}

The NBI method decomposes the MO problem into a series of SO problems and perform searches along homogeneously spaced parallel trajectories. The trajectories are defined to be normal to the simplex $H$ that connects the optimal solution for each performance metric $f^*_i$ (i.e. the extremities of $\partial F$), as shown in Figure 8.10.

The main steps in this decomposition strategy are developed next.

\textsuperscript{1}Contrary to MOEAs, single-objective EAs do not present these drawbacks related to the concept of Pareto-optimality. Thus, single-objectives EAs can be used in the frame-
8.2. PARETO-FRONT COMPUTATION BY NUMERICAL OPTIMIZATION

Figure 8.10: The NBI method in two and three dimensions.

(Step.1) Identification of individual minima: The extreme trade-offs are identified solving the optimization problem for each performance individually:

\[
\mathbf{f}^* = \min_{\mathbf{x}} f_i(\mathbf{x}) \quad \text{s.t.} \quad \begin{cases} 
  c_{eq}(\mathbf{x}) = 0 \\
  c_{ineq}(\mathbf{x}) \leq 0
\end{cases} \quad \text{for } i = 1, \ldots, n
\]  

(8.4)

For the NBI to cover the entire span of the PF, these performance vectors \( \mathbf{f}^* \) have to be the global minimum of each problem. Therefore, we recommend the use of hybrid algorithms (evolutionary + gradient methods) to solve the optimization problem (8.4). This avoids the need for providing a good starting point and prevents the optimization from becoming trapped in a local minimum.

(Step.2) Objective normalization: The following steps in the NBI method will involve the combination of all objectives. For the efficiency of the numerical optimization, it is important that all objectives are similarly scaled. The previous step has identified the minimum for each metric. Because of the contradictory nature of the objectives, the solution that minimizes one metric is likely to maximize another. Then, the minimum and maximum values observed in Step.1 can be used to normalize the objectives:

\[
f_i(\mathbf{x}) \leftarrow \frac{f_i(\mathbf{x}) - f_{i,\text{min}}}{f_{i,\text{max}} - f_{i,\text{min}}}
\]

(8.5)

(Step.3) Generation of the target trajectories: The simplex \( \mathcal{H} \) is the convex hull of the individual minima, i.e. the set of all points that are convex work as solvers for SO problems.
combinations of its vertices \( f_i \). With a matrix \( \Phi \) having the vectors \( f_i \) as its columns, this definition is formalized as:

\[
\mathcal{H} = \Phi \cdot \mathbf{w} : \mathbf{w} \in \mathbb{R}^n_+ , \quad \sum_{i=1}^{n} w_i = 1
\] (8.6)

\[
\Phi = \begin{bmatrix} f^{*1} & f^{*2} & \ldots & f^{*n} \end{bmatrix}
\] (8.7)

To generate an even spread of points in the PF, the NBI method discretizes \( \mathcal{H} \) uniformly by varying the weights \( w_i \) at fixed steps. Figure 8.11 gives an example for the case of three objectives where each component is discretized in 5 steps. In the common case where the same number of steps \( d \) is used for all \( n \) components, it can be shown that the generated set of vectors \( \mathcal{W} = \{w_1 \ldots w_k\} \) quickly grows with the number of objectives [SGA07]:

\[
k = \frac{(d + n - 1)!}{(d! \cdot (n - 1))}
\] (8.8)

This uniform discretization of the vectors \( \mathbf{w} \) relates to the field of Design of Experiments (DOE), where it is called a mixture design, and can also be interpreted from the point of view of geometry as a set of barycentric coordinates or as a triangulation since it establishes connectivity between neighbors.

Having a discrete set of points \( \Phi \mathbf{w} \), the definition of the trajectories is completed by computing the vector \( \mathbf{n} \) that is orthogonal to \( \mathcal{H} \). This vector is given by the coefficients of the hyperplane equation:

\[
a_1 f_1 + a_2 f_2 + \ldots + a_n f_n - a = 0 \quad \Rightarrow \quad \mathbf{n} = (a_1, a_2, \ldots, a_n)
\] (8.9)
8.2. PARETO-FRONT COMPUTATION BY NUMERICAL OPTIMIZATION

which is solved using Cramer’s rule for the known points $f^*$, giving the expression:

$$a = \det(\Phi)$$  \hspace{1cm} (8.10)

and $a_i$ equal to the determinant of $\Phi$ with the $ith$ column substituted by ones. The vector obtained through these calculations is normalized and, by convention, its sign is appropriately changed to make it point towards the origin using the formula:

$$\hat{n} \leftarrow -\hat{n} \cdot \text{sign}(\hat{n} \cdot 1)$$  \hspace{1cm} (8.11)

where the operator $\bullet$ denotes the scalar product and $1$ is the vector $[1...1]$ in $\mathbb{R}^n$. This calculation of the unit vector $\hat{n}$ differs from other works, that approximate it by $n = -\Phi \cdot 1$. Such an approximation does not impact the properties of the NBI method, but it is likely to be inaccurate for more than two dimensions (authors call it a quasi-normal vector [DD98, SGA07]). In this work, a geometrically correct definition of the normal is preferred because it benefits the subsequent interpolation task.

(Step.4) Formulation of the SO problems: For each discretized point $\Phi w$, the NBI problem is formulated as:

$$\max_{x, t} t \quad \text{s.t.:} \begin{cases} \Phi \cdot w + t \cdot \hat{n} = f(x) \\ c_{eq}(x) = 0, \ c_{ineq}(x) \leq 0 \end{cases}$$  \hspace{1cm} (8.12)

This states that the solution $f = f(x)$ is constrained to be at the tip of the vector $\Phi w + t \hat{n}$, while its (signed) distance $t$ to $\Phi w$ has to be maximized along the line $N$. Figure 8.12a depicts the elements in this formulation. It shows the case where the solution violates this equality constraint and must be improved in two directions to reduce the constraint violations $cv_1$ and $cv_2$.

Equality constraints are more difficult to satisfy than inequalities [LY08]. The practical resolution of problem (8.12), often replaces the vector equality by the inequality:
CHAPTER 8. PREDICTIVE HIERARCHICAL SYNTHESIS USING PARETO-FRONTIONS

Figure 8.12: Visual representation of different formulations of the BI problem. Each constraint violated by solution $\mathbf{f}$ is denoted by its magnitude $cv$.

$$\Phi \cdot \mathbf{w} + t \cdot \hat{\mathbf{n}} \geq f(\mathbf{x}) \quad (8.13)$$

This formulation is known as the Goal Attainment (GoAtt) problem. Under the assumption that $\partial F$ has no dominated solutions (monotone shape), then both formulations are equivalent. In the opposite case, depicted in Figure 8.12b, the GoAtt formulation may not return the expected trade-off when solved with local search methods. In the example, the inequality on objective $f_1$ is satisfied. The algorithm will move in the direction of reducing the constraint violation $cv_2$ and will not necessarily make the solution $\mathbf{f}$ intercept the normal line $N$, creating the paradoxical situation of being “stuck in (another) global minimum”.

In order to remedy this behavior, we propose to complement the GoAtt formulation with an additional constraint: $\mathbf{f}$ must lie within a distance $\varepsilon$ of the line $N$:

$$\text{dist}(\mathbf{f}, N) = \sqrt{\|\mathbf{f} - \Phi \mathbf{w}\|^2 - ((\mathbf{f} - \Phi \mathbf{w}) \cdot \hat{\mathbf{n}})^2} \leq \varepsilon \quad (8.14)$$

The penalty generated by a violation of this constraint (Figure 8.12c) allows the search to perform some hill climbing and go across concavities in $\partial F$. These hill climbing iterations are possible under the assumption that they reduce $cv_{dist}$. This assumption is true up to the moment where the climb direction (the tangent of $\partial F$) becomes parallel to $N$ and is consistent with the hypothesis in the NBI method. Any PF with a steeper curvature cannot be correctly discretized by NBI, because some trajectory would intercept $\partial F$ more than once. When compared with (8.12), see Figure 8.12a, this...
constraint has the benefit of being independent of the parameter $t$ which is being continuously changed during the optimization process. Of course, the same difficulties related to satisfying equality constraints will arise if $\varepsilon$ is set too small. From practical experience, $\varepsilon$ can be set as large as possible, as long as the cylinder that it describes does not intersect with neighbor trajectories. Considering the normalization of the objectives around the range $[0,1]$ (8.5), the trajectories are separated by a ball of radius $r = \sqrt{2}/(d - 1)$ in the bi-objective case (Figure 8.13). Even though this distance changes for higher dimensional cases according to the positions of $f^*_{ij}$, the choice of $\varepsilon = r/2$ should be adequate in most cases.

### 8.2.2 Methods for solving the constrained single-objective problems

Several works have successfully solved such constrained nonlinear optimization problems using Sequential Quadratic Programming (SQP) [BT95]. SQP methods solve local approximations of the optimization problem, with quadratic objective and linearized constraints combined in a merit function via penalty factors. For the specific solution of GoAtt problems, special treatment is given to the slack variable $t$, which is variable and objective at the same time and is not connected to the real design problem. As proposed by Brayton et al. [BDHV79], the objective $t$ should be left out of the merit function used in the inner iterations of the SQP method (called line search) and only the maximum constraint violation should be minimized (called minimax problem). This prevents the search to move in directions that improve $t$ but do not represent any improvement in bringing $f$ closer to the target trajectory.
The structure of the NBI method, with multiple SO problems to solve, allows for the use of some parallel strategies (Figure 8.14). The basic strategy, called jump start, consists of using the solution of an already solved problem as the starting point of a neighbor problem [DD98]. This is the approach implemented in this work, with the problems being solved from the extremes to the center of the PF. However, it is a pure local search strategy and may fail for the particular cases where the transition from the starting point to the target point requires the temporary, but large, violation of some design constraints.

Two alternative strategies that perform a global search are MOEA/D and Wavefront. MOEA/D [ZL07] manages multiple EAs in parallel. Each EA focuses on one trajectory and solutions are exchanged between neighbor populations. The Wavefront method [MGS07] follows the same idea but using SQP. If one problem converges prematurely, another one can be branched to continue the search.
8.3 Building predictive models

The previous section described the NBI method, which can provide an homogeneous discretization of the PF. According to the discretization step fixed by the designer, the method returns a set of $k$ optimal solutions $(x, f)$ approximating both the PF and the PS, as depicted in Figure 8.15.

Since the solutions represent distinct performance trade-offs, a one-to-one correspondence exists between points in the PF and points in the PS. A predictive model $G(\cdot)$ aims at providing a continuous representation of this correspondence, so that designers (or synthesis tools) can explore the entire span of the PF, selecting the most adequate trade-off and retrieving the associated sizing parameters immediately. According to the definitions in section 8.1.2, a predictive model is composed of three functions:

$$[f, x, v] = G(f^*) \equiv \begin{cases} f = G_f(f^*) & \text{(performance prediction)} \\ x = G_x(f^*) & \text{(parameter prediction)} \\ v = v(f^*) & \text{(domain of validity)} \end{cases} \quad (8.15)$$

$G_f(\cdot)$ reconstructs the PF by interpolating the performance vectors, $G_x(\cdot)$ establishes the correspondence between performance and sizing parameters and $v(\cdot)$ indicates whether the input to the model falls within the range covered by the PF.

In this section we detail the construction of these functions. The discussion focuses on two aspects: the choice of the independent variables for interpolation ($f^*$) and the construction of the signed-distance function $v(\cdot)$. We show that, for particular cases, it is not possible to parameterize $G(\cdot)$ as a func-
CHAPTER 8. PREDICTIVE HIERARCHICAL SYNTHESIS USING PARETO-FRONTS

\[ f = G_2(f_1) = \begin{bmatrix} f_1 \\ g_{22}(f_1) \end{bmatrix} \]

\[ x = G_1(f_1) = \begin{bmatrix} g_{11}(f_1) \\ g_{12}(f_1) \end{bmatrix} \]

\[ v(f_1) = \begin{cases} \leq 0 & \text{if } f_1 \in [f^-_1, f^+_1] \\ > 0 & \text{otherwise} \end{cases} \]

**Figure 8.16:** The three functions constituting a predictive model.

8.3.1 Basic structure of the predictive model

The three functions that compose a predictive model are illustrated in Figure 8.16. Although the illustration shows a bi-objective case, the following analysis applies for the general case of \( n \) performance metrics and \( m \) sizing parameters.

The PF is a hypersurface embedded in \( \mathbb{R}^n \). In most cases, it can be parameterized by computing one of the metrics as a function of the others (Figure 8.16a). The function \( G_f(\mathbf{f}^o) \), thus, takes as input a vector composed of \( n - 1 \) metrics:

\[ f = [f_1, f_2, \ldots, f_n] \Rightarrow \mathbf{f}^o = [f_1, f_2, \ldots, f_{n-1}] \quad (8.16) \]

and calculates the value of the remaining metric \( f_n \) by interpolating the discrete data points. The trade-off vector \( f \) with its \( n \) metrics is reconstituted
combining the input \( f^o \) and the predicted value \( f_n = g_{fn}(\cdot) \):

\[
\begin{bmatrix}
 f_1 \\
 \vdots \\
 f_{n-1} \\
 f_n
\end{bmatrix} = f = G_f(f^o) = \begin{bmatrix}
 f_1 \\
 \vdots \\
 f_{n-1} \\
 g_{fn}(f_1, \ldots, f_{n-1})
\end{bmatrix}
\] (8.17)

The function \( G_x(f^o) \) uses the same parameterization and interpolates the data points in the PS. An interpolation function is established for each design parameter in the vector \( x \) (Figure 8.16b). Using the same notation we have:

\[
\begin{bmatrix}
 x_1 \\
 \vdots \\
 x_m
\end{bmatrix} = x = G_x(f^o) = \begin{bmatrix}
 g_{x1}(f^o) \\
 \vdots \\
 g_{xm}(f^o)
\end{bmatrix}
\] (8.18)

The prediction performed by these functions is meaningful only if the input \( f^o \) lies within the region \( \Omega \) covered by the PF. Therefore we also have to build a classification function \( v(f^o) \) to distinguish between values inside and outside of \( \Omega \) (Figure 8.16c).

It is important to note that the construction of these three functions is completely dependent on the choice of the interpolation variables \( f^o \). According to the shape of the PF, not all choices are valid to reconstruct the discrete data. To exemplify this, let us consider the cases of a monotonic and a non-monotonic PF. If the PF is monotonic (Figure 8.17a), the relation between \( f_1 \) and \( f_2 \) is bijective and one can write either \( f_1 = g(f_2) \) or \( f_2 = g(f_1) \). A numerical difference can exist between these two alternatives, but the overall shape of the reconstructed curve is equivalent. However, if the PF is non-monotonic (Figure 8.17b), the choices are reduced. In the example, the PF can be expressed as a function of \( f_1 \), but not of \( f_2 \). It is also possible that some non-monotonic shapes (Figure 8.16c) do not admit any parameterization as a function of the performance metrics.

Fortunately, this difficulty or even impossibility to identify an adequate parameterization based on performance metrics can be addressed if we reconsider the way in which the data points have been obtained using the NBI
CHAPTER 8. PREDICTIVE HIERARCHICAL SYNTHESIS USING PARETO-FRONTS

Figure 8.17: The choice of the interpolation variables is limited by the shape of the PF.

Method. Indeed, the procedure that the NBI method uses to generate its parallel search directions defines important geometrical and topological relationships in the data. In the next two sections we explore these relations: first we introduce a change of variables allowing the predictive model to be constructed regardless of the shape of the PF returned by NBI. This change of variables, however, presents some drawbacks for the exploitation of the model. Therefore, we secondly describe a procedure to detect automatically which combinations of performance metrics $f^\diamond$ can provide a correct parameterization of data.

8.3.2 Change of variables

The discretization performed by the NBI method is based on two key geometrical constructions (i) the identification of a hyperplane $\mathcal{H}$ (more accurately a simplex) passing through the individual minima $f^\ast_i$ and (ii) the definition of a series of parallel search trajectories that are normal to the hyperplane. The hyperplane and its normal vector form an orthogonal system of coordinates, and the use of parallel trajectories enables the sampling of points in the PF for distinct coordinates in the hyperplane. The coordinates of the hyperplane are, hence, the natural choice to parameterize the PF and PS.

Figure 8.18 summarizes this reasoning and depicts the change of variables we propose to use in the predictive model. According to the coordinate system used in NBI, a performance vector $f \in \mathbb{R}^n$ can be expressed by (i) a vector $u \in \mathbb{R}^{n-1}$ indicating its coordinate in the hyperplane $\mathcal{H}$ and (ii) by a scalar $t$ indicating its distance to the hyperplane. This change of variables
8.3. BUILDING PREDICTIVE MODELS

\[
\begin{align*}
\mathbf{u} &= \Theta \mathbf{f} \\
\Theta &= \left[ \mathbf{b}_1 \mid \mathbf{b}_2 \mid \ldots \mid \mathbf{b}_{n-1} \mid \mathbf{n} \right]^T
\end{align*}
\]

(8.19)

where \( \Theta \) is an orthogonal matrix whose rows are the basis vectors of the new coordinate system. One of the basis vectors is \( \mathbf{n} \), the normal to \( \mathcal{H} \) that has been calculated by equations (8.9 - 8.11). This vector is placed in the last row of \( \Theta \) because we have defined the scalar \( t \) to be the last output of the transformation. Since \( \mathbf{n} \) has been computed as the exact normal to \( \mathcal{H} \) (not a quasi-normal approximation), another basis vector is given by:

\[
\mathbf{b}_1 = \mathbf{f}^{*1} - \mathbf{f}^{*2} \implies \mathbf{b}_1 \perp \mathbf{n}
\]

(8.20)

With these two basis vectors, the remaining orthonormal vectors \( \mathbf{b}_2, \ldots, \mathbf{b}_{n-1} \) are generated using the Gram-Schmidt orthogonalization process [Ant10].

By applying this transformation to the \( k \) performance vectors returned by the NBI optimization process, we get two sets of values:

\[
\{ \mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_k \} \xrightarrow{\Theta} \begin{cases} 
\{ \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k \} \\
\{ t_1, t_2, \ldots, t_k \}
\end{cases}
\]

(8.21)

\[\text{Figure 8.18: Change of variables: the parameterization is performed in the same coordinate system that the NBI method uses for sampling.}\]

\[\text{f} \rightarrow [\mathbf{u}, t] \text{ is a simple rotation of axis and can be expressed by a linear transformation}^2:\]

\[
\begin{bmatrix}
\mathbf{u} \\
t
\end{bmatrix} = \Theta \mathbf{f}
\]

(8.19)
Now, there is only one choice to parameterize the PF: to interpolate the values of \( t \) as a function of \( u \), i.e. \( t = g_t(u) \). The performance prediction function \( G_f(f^\circ) \) becomes \( G_f(u) \). We write it similarly to (8.17), but adding the inverse matrix \( \Theta^{-1} \) to map the predicted values \([u, t] \rightarrow f\):

\[
f = G_f(u) = \Theta^{-1} \begin{bmatrix} u \\ g_t(u) \end{bmatrix}
\]

Due to the one-to-one correspondence between PF and PS, if \( u \) parameterizes the PF, it can also serve to parameterize the PS. Thus, the function predicting sizing parameters maintains the same format:

\[
x = G_x(u) = \begin{bmatrix} g_{x1}(u) \\ \vdots \\ g_{xm}(u) \end{bmatrix}
\]

Similarly, the validity function becomes \( v(u) \), with the domain \( \Omega \) being the projection of the PF onto the hyperplane \( \mathcal{H} \).

The proposed change of variables exploiting the geometrical relations of the NBI method provides a general solution to the problem of finding an adequate parameterization for the predictive model. Any shape of PF that can be detected by the NBI method can be parameterized by the vector \( u \). Additionally, the change of variables can potentially benefit the robustness of the interpolation. Indeed, since the same system of coordinates used by the NBI method is used for interpolation, the property of equally spaced samples in the performance space is preserved. This can be noticed in Figure 8.19, where a curved 3D PF is projected onto the three planes corresponding to the performance metrics and onto the plane \( \mathcal{H} \) used by NBI. The projection onto \( \mathcal{H} \) preserves the regular spacing of the samples (compare with Figure 8.10), whereas the projection onto the other planes becomes distorted according to the curvature of the PF.

However this change of variables presents a drawback: the input of the model \( G(u) \) has no physical meaning, contrary to \( G(f^\circ) \). For the synthesis tool manipulating the model, this presents no problem, since the tool will iteratively vary the input to achieve the desired output. However, for a human designer
Figure 8.19: Projections of the PF onto the normal sampling plane and onto the planes $\Pi_{xy}$, $\Pi_{xz}$ and $\Pi_{yz}$. Due to the curvature of the PF, the projections get distorted and can fold onto themselves. Interactive plot when viewed with Acrobat Reader 9.
CHAPTER 8. PREDICTIVE HIERARCHICAL SYNTHESIS USING
PARETO-FRONTs

Figure 8.20: The mixed use of $G(f^\circ)$ and $G(u)$ in the synthesis flow.

this makes the model unusable. A designer needs, instead, to have a model $G(f^\circ)$ where he/she can manipulate performance metrics explicitly. These points are summarized in Table 8.1.

| $G(u)$                  | ⊗ Uniform distribution of points |
|                        | ⊗ Always parameterizable          |
|                        | ⊗ Variable with no physical meaning |
|                        | ⊗ Can be handled by optimization tools |

| $G(f^\circ)$          | ⊗ Physical meaning, designers can directly query the model |
|                       | ⊗ Distribution of points distorted by projection |
|                       | ⊗ Not always parameterizable          |
|                       | ⊗ Only necessary for interaction with designer |

Upon consideration, it seems necessary to build both alternatives: $G(u)$ to be used in all automated processes and $G(f^\circ)$ reserved for the interaction with the designer. If both representations have to be combined in the same task, such as the TD synthesis process (Figure 8.20), the transformation $\Theta$ is used to adapt the inputs as necessary.

Nevertheless, with the need to provide a predictive model $G(f^\circ)$, we are obliged to tackle the problem of identifying which set of variables $f^\circ$ forms an admissible parameterization of the PF and PS.

8.3.3 Identification of admissible parameterizations for $G(f^\circ)$

For any block with more than one performance metric, there are multiple choices to parameterize the predictive model $G(f^\circ)$. For example, with three metrics we may write $G(f_1, f_2)$, $G(f_1, f_3)$ or $G(f_2, f_3)$. However, as previ-
8.3. BUILDING PREDICTIVE MODELS

ously discussed, some combinations may not provide a correct reconstruction of the data set and we have to provide a systematic method to detect the right ones.

To perform this task, we will use the parameterization $G(\mathbf{u})$ as a reference, since it reconstructs the PF in a way that is consistent with the sampling performed by NBI. Hence, for a parameterization $f^o$ to be admissible, there must be a one-to-one correspondence between $G(f^o)$ and $G(\mathbf{u})$ or, in a simpler way, between $f^o$ and $\mathbf{u}$.

To verify this condition let us consider the mapping $G_f : \mathbf{u} \rightarrow f$ in equation (8.22). Assuming that $f^o$ excludes the $n$th metric, we rewrite the mapping in the form $H : \mathbf{u} \rightarrow f^o$:

$$
\begin{bmatrix}
 f_1 \\
 \vdots \\
 f_{n-1}
\end{bmatrix} = f^o = H(\mathbf{u}) = \Theta_n^{-1}
\begin{bmatrix}
 \mathbf{u} \\
 g_t(\mathbf{u})
\end{bmatrix}
$$

where $\Theta_n$ excludes the $n$th line of matrix $\Theta$. The vector $f^o$ is admissible if the transformation function $H(\cdot)$ is bijective. This property can be tested calculating the determinant of the Jacobian of $H(\cdot)$:

$$
|J_H(\mathbf{u})| = \det
\begin{bmatrix}
 \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_{n-1}} \\
 \vdots & \ddots & \vdots \\
 \frac{\partial f_{n-1}}{\partial u_1} & \cdots & \frac{\partial f_{n-1}}{\partial u_{n-1}}
\end{bmatrix}
$$

and verifying that it has positive sign over the entire domain $\Omega$ (refer to the inverse function theorem [AZ10]).

From the standpoint of numerical implementation, it is clear that we cannot sweep every point in $\Omega$ to evaluate the sign of the Jacobian. In fact, the only points that need to be considered are those in the set $\{f_1, f_2 \ldots f_k\}$ returned by NBI. With this discretization, the only assumption that we can make on $g_t(\mathbf{u})$, and consequently on $H(\cdot)$, is that it is piecewise linear. Thus, the PF is approximated by a collection of patches. This greatly simplifies the computation of the Jacobian and, in fact, allows us to give an alternative interpretation using concepts from the mathematical fields of topology and geometry.
According to the dimension $n$ of the PF, the patches are line segments, triangles, tetrahedrons, etc. Each of these simplicial forms is orientable (Figure 8.21). Orientation can be positive or negative (with respect to an arbitrary convention). For the example of a line segment, its orientation $O$ is simply calculated by $O = \text{sign}(x_1 - x_0)$. In the general case, the orientation of a $N$-simplex is given by the determinant of a matrix composed of its $N$ vertices (see the computation of the oriented volume of a simplex [Han94]):

$$O_N = \text{sign} \left( \det \begin{bmatrix}
(x_1 - x_0) & (x_2 - x_0) & \cdots & (x_N - x_0) \\
(y_1 - y_0) & (y_2 - y_0) & \cdots & (y_N - y_0) \\
\vdots & \vdots & \ddots & \vdots \\
(w_1 - w_0) & (w_2 - w_0) & \cdots & (w_N - w_0)
\end{bmatrix} \right) \quad (8.26)$$

Now, the field of topology defines that a transformation is bijective if and only if it preserves (or reverses) the orientation of all patches of a surface [PS06, Flo97] (a so-called homeomorphism). To transpose this concept to our problem, let us consider Figure 8.22. We define the orientation of each patch of the PF on the axis $u$ and then compare it with the orientation computed by projecting the patch onto the axes $f_1$ and $f_2$. The projection onto $f_2$ preserves all orientations, hence we can write $G(f_2)$. Contrary to this, the projection onto $f_1$ reverses the orientation of one patch, showing that there is a one-to-many correspondence between $f_1$ and $u$. Thus, $f_1$ is not admissible to parameterize the predictive model. In conclusion, this test identifies the projections onto which the PF folds onto itself, making it impossible to parameterize. Figure 8.23 shows its application for the 3D case (reconsidering the case on Figure 8.19). The projection of the surface onto the plane $\Pi_{f_1f_3}$ folds onto itself, causing a change of orientation. On the other hand, the projection onto $\Pi_{f_1f_2}$ preserves the orientation everywhere,
letting one write $G(u_1, u_2)$ or $G(f_1, f_2)$.

The numerical implementation of the test compares the orientations patch by patch. The first two patches compared define whether the transformation $u \rightarrow f^v$ preserves or reverses orientation. If any of the following patches violates this definition (i.e. a second inversion), then the test stops immediately, excluding the given $f^v$ (e.g. $[f_1, f_3]$) from the possible choices for parameterization.

The definition of the patches (lines, triangles, etc) implies the determination of connections between vertices. This can be done in two ways: either using the Delaunay triangulation method to identify and connect the points $\{u_1, \ldots, u_k\}$ to their closest neighbors [GO04]; or using the triangulation that is implicitly defined by the set of weighting vectors $W$ of the NBI method (see Figure 8.11).

To conclude, it is important to understand that this test does not extend to any dimension. As with many problems in topology, the homeomorphism problem can be undecidable for dimensions $n > 3$ [Mar58]. Simply put, the sign of the determinant in (8.26) alternates if two columns of the matrix are permuted. If the matrix has more than three columns, we can permute multiple columns at the same time: changing the order of the vertices, but preserving the sign of the determinant.
8.3.4 Interpolation with radial basis functions

We have discussed and presented solutions to identify the correct choices of parameters for the predictive model $G(\cdot)$, but did not give details on the interpolation method used to build the functions $g_{f_1}(\cdot), g_{t_1}(\cdot), \ldots, g_{x_m}(\cdot)$ interpolating the scattered data $\{f_1, \ldots, f_k\}$ and $\{x_1, \ldots, x_k\}$. Among the various existing interpolation techniques, we briefly describe one known as Radial Basis Function (RBF) interpolation. RBF interpolation has the advantage of being easy to implement and applicable to unorganized data (i.e. it does not need the data to fit into a rectangular grid).

To avoid confusion between inputs ($u$ or $f$) and outputs ($f_i$, $x_j$ or $t$) of the interpolation functions, let us simplify the notation and refer to any input as $y$ and to any output as $z$.

A RBF is a symmetrical function $\phi(y)$; one whose value depends only on the distance (radius) from the origin, i.e. $\phi(y) = \phi(||y||)$. Some popular basis functions are listed in Table 8.2. The principle of RBF interpolation is to approximate the output as a weighted sum of basis functions:

$$z = g(y) = \sum_{j=1}^{k} d_j \phi(||y - y_j||) \quad (8.27)$$

Each of the $j$ functions in the sum is centered at one of the sampled points.
Table 8.2: Typical functions for RBF interpolation with $r = \|y - y_i\|

<table>
<thead>
<tr>
<th>Function</th>
<th>$r^2 \ln(r)$</th>
<th>$\sqrt{1 + r^2}$</th>
<th>$(1 - r)^3[3r + 1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>$e^{-r^2}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thin-plate spline</td>
<td>$r^2 \ln(r)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiquadric</td>
<td>$\sqrt{1 + r^2}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wendland</td>
<td>$(1 - r)^3[3r + 1]$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\{y_1 \ldots y_k\} and has a weight $d_j$. An example is shown in Figure 8.24, where a sine curve is reconstructed as a series of Gaussian functions of different amplitudes. The Gaussian function vanishes when far from its center point, creating a limited zone of influence for each sample (called the support of the function). Like the Gaussian, Wendland’s function also vanishes [Wen95]. These are referred as “compactly supported” RBFs.

The weights $d_j$ in 8.27 are calculated once by solving the following linear system for the known values of $z$:

$$
\begin{bmatrix}
\phi_{11} & \phi_{12} & \ldots & \phi_{1k} \\
\phi_{21} & \phi_{22} & \ldots & \phi_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
\phi_{k1} & \phi_{k2} & \ldots & \phi_{kk}
\end{bmatrix}
\begin{bmatrix}
d_1 \\
d_2 \\
\vdots \\
d_k
\end{bmatrix}
=
\begin{bmatrix}
z_1 \\
z_2 \\
\vdots \\
z_k
\end{bmatrix}
$$

(8.28)

where $\phi_{ij} = \phi(\|y_i - y_j\|)$. Due to the symmetry of $\phi(\cdot)$, the system is symmetric and positive semi-definite. For very large systems (e.g., medical imaging with $>100k$ samples [SSB05]), compactly supported RBFs are interesting because they make the linear system sparse. However, in the context of this work the number of points is much smaller, probably not exceeding the hundred, and there is no computational penalty for using functions with “infinite support” like the Thin-plate spline. Additionally, one can combine RBF interpolation with polynomial approximation so that the polynomial fits the global shape of the surface and the RBFs reconstruct local features.

8.3.5 Approximation of the validity domain $\Omega$

The domain of validity of the interpolation functions is given by the projection of the PF onto the plane used for parameterization ($u$ or $f^o$). In the case of a bi-objective problem it is a simple identification of box constraints (upper and lower bounds) for a single input variable ($u_1^L \leq u_1 \leq u_1^U$). But for higher dimensions, $\Omega$ is a closed region of arbitrary shape (convex or
Thus, in addition to the box constraints, one has to identify its boundary \( \partial \Omega \) and build a classification function \( v(u) \) to distinguish points inside \( (v \leq 0) \) and outside the domain \( (v > 0) \).

One approach is to express \( \Omega \) as a set of linear constraints. Figure 8.25 depicts the process: the scattered data with an arbitrary boundary \( \partial \Omega \) is triangulated to extract its convex hull [BDH96]. The triangle faces belonging to the convex hull define a series of lines (hyperplanes) delimiting the region. Individually, the equation of each line separates two half-spaces:

\[
\mathbf{a}^\top \mathbf{u} - b < 0 \quad \text{and} \quad \mathbf{a}^\top \mathbf{u} - b > 0
\]

The region \( \Omega \) is delimited by the intersection of the half-spaces that, by convention, have negative sign\(^3\). Using a matrix notation to evaluate all equations, we have a criterion to classify interior points and, therefore, an expression for \( v(u) \):

\[
\text{interior points: } \mathbf{A} \mathbf{u} - \mathbf{b} < 0 \quad \implies \quad v(\mathbf{u}) = \max(\mathbf{A} \mathbf{u} - \mathbf{b})
\]

\(^3\)To make all equations comply with the convention, a test of (8.29) against a known interior point is used to determine the adequate sign of the coefficients \( \mathbf{a} \) and \( b \).
8.3. BUILDING PREDICTIVE MODELS

This method is simple and extends to any dimension. However, the intersection of half-spaces can only represent convex boundaries. Consequently, if the domain $\Omega$ is concave, the linear constraints end up defining a larger region.

To represent arbitrary concave boundaries, one has to resort to more sophisticated methods. A first step is to identify the points in the concave hull using an appropriate algorithm, such as Power Crust [ACK01] or $\alpha$-shapes [HE94]. The concave boundary can, then, be approximated in different ways. One can use “polygon-to-function” methods [PSS96, EI09], which make use of R-functions [Sha91] to perform boolean operations on real-valued functions, creating unions and intersections of sub-regions to enclose $\Omega$ tightly. Although powerful and precise, the implementation of these methods is complex.

An alternative, vastly used in 3D imaging and typically referred to as “implicit functions”, is to define the function $v(\mathbf{u})$ describing the closed region $\Omega$ (thus the term “implicit”) using interpolation methods for unstructured point sets, such as RBF interpolation. A comprehensive survey of these methods is given by Chang [Cha07]. Most authors treat the problem of reconstructing scanned objects (e.g. statues, bones) for which they have sampled a set of boundary points (i.e. $v(\cdot) = 0$) and the normal to the surface at each point [LK04]. The normal pointing towards the outside is used to constrain the interpolation, orienting the sign of the function $v(\cdot)$. In our case, however, we do not have information on the normals. Instead, we have samples both on the boundary and at the interior of $\Omega$, i.e. $v(\cdot) < 0$. To construct $v(\cdot)$, it is sufficient to assign to each sample $\mathbf{u}_k$ an approximate distance to the boundary and interpolate the points with RBFs.

In this work, we propose a very simple approximation of the distance that is based on the weighting vectors $\mathbf{w}_k$ used in the NBI method. The value of each component $[w_1, w_2, \ldots, w_n]$ of $\mathbf{w}$ indicates a distance to one of the boundaries, i.e. the ends of the line segment or the faces of the triangle in Figure 8.26a. Thus, we assign, for each sample point $\mathbf{u}_k$, the shortest distance to the boundary computed based on its corresponding weighting vector $\mathbf{w}_k$ (Figure 8.26b):

$$v(\mathbf{u}_k) = -\min(\mathbf{w}_k)$$ (8.31)
CHAPTER 8. PREDICTIVE HIERARCHICAL SYNTHESIS USING PARETO-FRONS

(a) The weighting vectors $w = [w_1 \ldots w_n]$ used in the NBI method. For convenience, the values of its components are considered as integers.

(b) Reuse of the information of the weighting vectors to assign an approximate distance to the boundary. For each discretized point $u_j$, the distance value is set according to its associated weighting vector: $v(u_j) = -\min(w_j)$.

(c) Interpolation of the points to construct the classification function $v(u)$.

Figure 8.26: Process of construction of the validity function $v(\cdot)$ using RBF interpolation. For a 2D case, this is not necessary, but its illustration facilitates the understanding. The 3D plot (c) is interactive when viewed with Acrobat Reader 9.
The negative sign accounts for the adopted convention of \( v(\cdot) \). Of course, as the projection of the PF onto a plane can deform the original triangulation, the proposed approximation of the distance is rough. This is, however, acceptable since we are more interested in the sign of \( v(\cdot) \) than in its value. Figure 8.26c shows the result of the interpolation using Thin-Plate Spline (TPS) as basis function. Two remarks on this method are that:

1. The shape of \( v(\cdot) \) is only constrained by the interior points. Exterior to \( \Omega \) there is no other control point for interpolation and \( v(\cdot) \) simply follows the trend imposed by the interior points. To maintain this trend everywhere within the region defined by the box constraints \([u^-, u^+]\), it seems interesting to use basis function with a large support (TPS has infinite support).

2. The interpolation functions are smooth and may not reproduce sharp features accurately, such as the three corners of the triangle. Solutions to model sharp features more precisely include increasing the number of points near the corners \([OBA^+05]\) or considering the function by pieces \([FCOS05]\).

We conclude here the discussion on the mathematical formulations and geometrical constructions that support the construction of all the three functions of a predictive model \( G(\cdot) \), namely \( G_f(\cdot) \), \( G_x(\cdot) \) and \( v(\cdot) \).

8.4 Accounting for environment parameters

In the introductory chapter, we have presented the performance of a block as a function not only of design parameters \( x \), but also of the environment \( e \), i.e. \( f = F(x, e) \). To simplify the discussion on the predictive models, we have considered the environment to be fixed prior to the PF characterization. However, as the objective of the framework is to provide a library of predictive models capable of adapting to evolutions in the requirements and of being reused for different applications, it becomes necessary to account for changing environment conditions. We discuss, in this section, the additional steps in the characterization process to accomplish this.

The design of a block has to adapt to (i.e. be optimized for) the imposed environment. Thus, environment parameters \( e \), contrary to \( x \), are not modified.
by the optimization process. To include the impact of a changing environment in the predictive models, we have to characterize the Pareto trade-off curves of a block for a limited number of scenarios \( S = \{e_1, e_2, \ldots, e_s\} \) and then interpolate them to provide a continuous representation. Figure 8.27 illustrates the process for a single parameter \( e_1 \) whose range has been discretized by three values. For the interpolation process, \( e_1 \) becomes a new axis. The range covered by \( e_1 \) is included in the definition of the valid domain \( \Omega \) in the same way as performance metrics. Thus, the interface of the predictive model remains unchanged:

\[
[f, \mathbf{x}, v] = G(f^0, \mathbf{e})
\]  

(8.32)

Still, a particularity arises from the fact of having executed the characterization process multiple times: each run of the NBI process follows a different normal vector (Figure 8.28a). Therefore, to apply the change of variables we have proposed, one has to consider an average normal direction to establish the system of coordinates \( \mathbf{u} \times \mathbf{t} \) (Figure 8.28b).
For the case of multiple parameters, i.e. \( e \in \mathbb{R}^q \), one may apply a factorial design or other DOE approach [BD87] to generate a structured set of scenarios \( S \) to be evaluated. However, it is clear that the so-called dimensionality curse comes into action here, limiting the number of parameters \( q \) that can be handled in affordable runtime. Nevertheless, being able to characterize the impact of only one or two key environment parameters, such as input, load or temperature conditions, can already greatly extend the reusability of a model.

### 8.5 Summary

In this chapter we have presented a framework for the hierarchical design of heterogeneous systems based on Pareto Fronts. The framework extends the state of the art by introducing the concept of a predictive model that interpolates both the PF and the PS. The designer can, thus, explore the performance trade-off curves, select the point that suits its requirements and get an immediate estimate of the sizing parameters for the block.

The predictive models integrate seamlessly in a typical optimization cycle, enabling design automation. Moreover, since they operate on a set of stored design points, they are self-contained and completely abstract the semantics of the simulation model that they represent. Predictive models are, thus, an effective way to exchange design information and support hierarchical design of mixed-domain systems.

A second part of the chapter has been dedicated to a detailed discussion of the optimization and interpolation processes required to build a predictive model. We have assumed that PFs can be non-convex and present some non-dominated regions. Therefore, the NBI method has been used to manage the optimization process and return an even discretization of the trade-off surface. The discretized PF and PS were reconstructed using radial basis functions. Non-dominated regions in the PF, instead of being discarded as discontinuities, have also been modeled as a means to provide a continuous representation over the entire span of the PF. Moreover, we have extensively discussed the process of parameterization of the curves, relating it to the NBI algorithm and providing techniques to ensure that the predictive model can reconstruct the information correctly.
9 Experimental results

To verify the benefits of the predictive synthesis flow and validate its software implementation, three test cases are studied:

- An optical data link decomposed into three blocks;
- An Operational Transconductance Amplifier (OTA);
- A synthetic application with dominated solutions in the PF.

Each application focuses on a specific aspect of the predictive synthesis flow to facilitate the analysis of our test results.

9.1 Implementation and experimental method

9.1.1 Implementation of the framework

The framework is implemented in MATLAB and Java. The methods of sections 8.2 and 8.3 that cover optimization and interpolation are written in the MATLAB language.

The interface with external simulators is implemented in Java. Java provides more powerful mechanisms for Inter-Process Communication (IPC) than MATLAB, such as pipes, sockets and threads [Fla05]. MATLAB natively supports the execution of Java code, the integration of both languages is seamless and efficient. At the current state of the platform, the following languages and tools are supported for modeling and simulation:

- MATLAB / Simulink;
- C++ / SystemC [ACC];
- Cadence Spectre [CAD] (with support for Verilog-AMS);
- Mentor Graphics Eldo [MEN];
- SpiceOpus, an open-source spice-like simulator [OPU].
9.1. IMPLEMENTATION AND EXPERIMENTAL METHOD

9.1.2 Methodology

For all test cases, the objective of the experiments is to validate the capacity of the predictive synthesis to correctly size the system to meet the required performance trade-off. Therefore a complete cycle of synthesis and validation is carried out. To quantify prediction quality, performance values issued by the prediction synthesis flow are compared to simulation results (Figure 9.1). The difference between predicted and simulated values serves to calculate a prediction error for each performance metric:

\[
\text{prediction error (\%)} = \frac{\text{simulated} - \text{predicted}}{\text{predicted}} \times 100
\]  

(9.1)

By repeating this process for multiple requirements within the feasible region, we get a statistical measure of the error in the form of histograms.

This experimental methodology is applied to three test cases:

**Optical data link**: The first test case is a demonstration of the hierarchical design flow in a problem mixing the electrical and optical domains. Components are characterized by low-level simulations and combined in a SystemC model of an optical NoC;

**OTA**: The design of a transconductance amplifier with Miller compensation allows to test the accuracy of the predictive model in a problem with a larger number of variables and performance metrics;

**PF with dominated points**: A synthetic hierarchical application based on analytical expressions is built to produce a non-monotonic PF. We investigate the impact of this type of PF in the BU characterization and in the TD synthesis steps.
9.2 Data link for optical NoC application

9.2.1 Description

The integration of passive and active photonic devices compatible with CMOS technology allows to replace electrical interconnections by high speed optical links. This addresses the increasing demand for bandwidth [PP10]. Optical NoCs (ONoC) can connect multi-processor architectures offering low latency and, thanks to wavelength routing, low contention [BGB+07]. As shown in Figure 9.2, the ONoC basically consists of opto-electrical interfaces (transmission and reception) and of a passive optical router (λ-router). Our research group previously studied this basic optical link [BGB+07, AOD+09, AOS10], a suitable example to illustrate the handling of hierarchy and heterogeneity in the framework developed in this work.

Figure 9.3 shows the hierarchy of predictive models that will be built in this experiments. On the left are the inputs of the predictive model and on the right the complete list of performance metrics they compute. According to the relationship between hierarchical levels, the inputs to the low-level blocks are the design variables at system-level.

The design challenge at the optical link level consists of finding a good trade-off between power consumption, maximum data rate, and bit error rate (BER). Low BER is achieved by increasing the transmitted signal power to counterbalance losses in the router and photodetector. At higher data rates, however, further improvement is limited by the growing noise of the output’s transimpedance amplifier (TIA).

On the transmission side, a binary electrical signal is converted into an optical signal by a laser. The laser is driven by a CMOS circuit, which provides

\[ \lambda \text{-router} \]
9.2. DATA LINK FOR OPTICAL NOC APPLICATION

Figure 9.3: Optical link predictive model and its hierarchy

A constant bias current ($I_{\text{bias}}$) and modulates the logical ‘0’$\text{’}$s and ‘1’$\text{’}$s with an incremental modulation current ($I_{\text{mod}}$). The conversion efficiency of the laser (eff), expressed in W/A (optical power/electrical current), can be controlled by modifying the structural properties of the laser. However, an increase in conversion efficiency also entails an increase in the threshold current ($I_{\text{th}}$) to achieve the laser effect. The driver circuit must, therefore, provide a bias current superior to the laser threshold ($I_{\text{bias}} > I_{\text{th}}$).

On the reception side, the signal must be amplified to recover from the attenuation and noise due to the router and photodetector elements. The router and photodetector noise are modeled at system-level, as environment parameters set by the designer. The design parameter for the reception side is the gain of the amplifier circuit.

In order to perform such trade-off analysis, four simulation models were used to represent the hierarchy of the system:

- A SystemC model of the entire optical link [BGB+07, AOS10]. It includes the behavior of the active components and of the A-router;
- SPICE-like electrical models for the driver and TIA circuits using Cadence Spectre;
- VISTAS [JEB03], a spatio-temporal model of a Vertical Cavity Surface Emitting Laser (VCSEL) implemented in MATLAB.

9.2.2 Results

In a first step, the predictive models of the low-level blocks are generated using the NBI method. Table 9.1 summarizes the metrics considered for the characterization. Table 9.2 lists the nature of the design variables, the
number of points to discretize the PF and the total execution runtime of the characterization process. The trade-off curves for each component appear in Figure 9.4. This setup is commented in the next paragraphs.

The driver circuit in our system can be split into two independent structures: bias and modulation. Treating these independently, yields a 2D PF between current-power with 10 points for each structure. Assembling them together leads to the “rectangular” PF in the figure.

The laser component is considered to have a single design parameter: the reflectivity of the Bragg mirrors in its cavity (varying from 0.94 to 0.98, according to the parameters in the VISTAS model). Consequently, the trade-off curve is computed by a simple sweep of this parameter.

Finally, for the TIA circuit, we consider the gain-bandwidth trade-off. Minimum bounds are set for each of these metrics in order to restrict the search space. In addition, we consider the Input Refered Noise (IRN) and power as para-functional performance metrics. This means they are not considered for optimization, but are also evaluated and interpolated by the predictive model.

<table>
<thead>
<tr>
<th>Trade-off metrics</th>
<th>Bounds and remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver</td>
<td>maximize bias current</td>
</tr>
<tr>
<td></td>
<td>$500 \mu A &lt; I_{bias} &lt; 1.8mA$</td>
</tr>
<tr>
<td></td>
<td>maximize modulation current</td>
</tr>
<tr>
<td></td>
<td>$250 \mu A &lt; I_{mod} &lt; 1.8mA$</td>
</tr>
<tr>
<td></td>
<td>minimize power</td>
</tr>
<tr>
<td>Laser</td>
<td>maximize efficiency</td>
</tr>
<tr>
<td></td>
<td>limited by parameter bounds</td>
</tr>
<tr>
<td></td>
<td>minimize threshold current</td>
</tr>
<tr>
<td>TIA</td>
<td>maximize transimpedance gain</td>
</tr>
<tr>
<td></td>
<td>$&gt;1k\Omega$</td>
</tr>
<tr>
<td></td>
<td>maximize bandwidth</td>
</tr>
<tr>
<td></td>
<td>$&gt;1GHz$</td>
</tr>
</tbody>
</table>
Table 9.2: Characterization of the components of the optical link

<table>
<thead>
<tr>
<th>Component</th>
<th>Design parameters (No.)</th>
<th>No. of points</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver</td>
<td>Transistor widths (4)</td>
<td>10 × 10</td>
<td>6 min</td>
</tr>
<tr>
<td>Laser</td>
<td>Reflectivity of the cavity mirror (1)</td>
<td>9</td>
<td>30 min</td>
</tr>
<tr>
<td>TIA</td>
<td>Transistor widths (3)</td>
<td>10</td>
<td>23 min</td>
</tr>
</tbody>
</table>

To characterize the trade-offs at system-level, we use the optimization scheme of Figure 8.6 to combine the SystemC model with the predictive models of the blocks. A constraint of BER < $10^{-20}$ was imposed to set a limit in the exploration of the optical link PF. The resulting trade-off surface, involving power, data rate and BER is shown in Figure 9.5 as a contour plot\(^1\), while the corresponding sizing parameters (i.e. the PS) are shown in Figure 9.6. The maximum data rate and the BER can be improved together at the cost of increasing power up to 3.2Gb/s. From that point onwards, an increase in the data rate causes a worsening of the BER, while still requiring more power. This is mainly due to the growing noise and decreasing gain in the TIA.

We compare now the process of building the high level PF by optimization proposed in this work, with the MUBU approach [GME05]. Results for both approaches are reported in Table 9.3. In our approach, 45 optimization problems were solved to discretize the PF. MUBU does not use optimization, but combines the discrete points we have for the components (i.e. $10 \times 10 \times 9 \times 10$). Among the design points returned by MUBU, less than 3% did respect the constraint on the BER and 0.15% where non-dominated solutions. This shows that MUBU is not faster than an approach that re-

\(^1\)The contour plot covers the convex hull of the points. It is noticeable, however, that the region delimited by the points is concave.

Figure 9.5: Contour plot for the Pareto Front of the optical link
requires an optimization cycle and that the solutions obtained by the simple combination of Pareto-optimal points are not necessarily optimal.

In order to make the predictive model more flexible and stress our interpolation methods, we have repeated the characterization process of the optical link for another environment condition. The results presented previously were obtained considering a photodetector with responsivity of 0.5 A/W. We have repeated it for 0.4 A/W, a worse condition. The resulting predictive model interpolates between values of BER, data rate and responsivity. The domain of validity $\Omega$ for this model is a 3D volume presented in Figure 9.7. The input values lying at the interior of the volume are valid. It can be

![Figure 9.6: Interpolation of the optical link Pareto Set](image)

### Table 9.3: Comparison between the construction of high-level predictive models and the MUBU approach.

<table>
<thead>
<tr>
<th>High-level predictive model built with optimization</th>
<th>High-level PF built with the MUBU approach [GME05]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of optimization problems 45</td>
<td>Designs to simulate at system level 9000</td>
</tr>
<tr>
<td>Number of simulations required 2038</td>
<td>Designs meeting system constraints 288</td>
</tr>
<tr>
<td>Pareto optimal solutions 45</td>
<td>Pareto optimal solutions 14</td>
</tr>
<tr>
<td></td>
<td>9 minutes</td>
</tr>
<tr>
<td></td>
<td>22 minutes</td>
</tr>
<tr>
<td></td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>–</td>
</tr>
</tbody>
</table>
observed that the use of RBF interpolation is not able to fit the curve to the sharp edges of the PF and thus excludes the extreme trade-offs from \( \Omega \).

To complete the analysis, the TD synthesis process is considered. Our statistical analysis was performed using 2000 points randomly generated inside the volume of Figure 9.7. Predicted values of performance and sizing of the individual components were generated in less than 10 seconds. These sized designs were sent to the simulators for BU validation. The histograms of the prediction error are shown in Figure 9.8. For all three metrics, the error is below 2\% and follows a normal distribution centered at zero.

This experiment covered all steps in the hierarchical synthesis process. It validated the capacity of the predictive approach to provide an adequate sizing of the blocks of a system. Additionally, we showed that the characterization of high-level PFs using optimization can be more efficient than the MUBU approach.
9.3 Operational Transconductance Amplifier (OTA)

9.3.1 Description

The aim of this experiment is to verify how the predictive approach handles complex analog circuits. We consider here the design of an OTA with Miller compensation (Figure 9.9) and investigate the trade-off between power and gain-bandwidth product (GBW). This trade-off is characterized for multiple values of the loading capacitance $C_L$, leading to a flexible model of the amplifier.

Figure 9.10 summarizes the inputs and outputs of the predictive model. There is no hierarchy in the problem, but a large number of performance metrics and sizing parameters to be predicted. The model is parameterized by GBW and $C_L$. Besides power, there are several para-functional metrics to be predicted. The number of variables is 7: 5 transistors widths, the bias voltage and the Miller capacitance (symmetrical transistors like M1a and
9.3. OPERATIONAL TRANSCONDUCTANCE AMPLIFIER (OTA)

Table 9.4: Setup for the characterization of the OTA design trade-offs

<table>
<thead>
<tr>
<th>Trade-off</th>
<th>Gain-Bandwidth (GBW)</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>DC gain</td>
<td>maximize</td>
</tr>
<tr>
<td></td>
<td>Slew rate</td>
<td>minimize</td>
</tr>
<tr>
<td></td>
<td>Phase margin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DC output</td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>$V_{dc}$</td>
<td>3.3V</td>
</tr>
<tr>
<td></td>
<td>$C_L$</td>
<td>[0.5, 1.0, 1.2, 1.5] pF</td>
</tr>
</tbody>
</table>

Milb have the same width). The characterization setup is listed in Table 9.4. It indicates constraints on the para-functional metrics and gives a discrete set of values for the load capacitance $C_L$.

9.3.2 Results

Using the same problem setup, the OTA is characterized for two technology processes: AMS 0.35µm [AMS] simulated with Spectre and OnSemi 0.5µm (distributed by MOSIS [MOS]) simulated with SpiceOpus. The characterization step considers 15 points for each PF and 4 values for $C_L$ (total of 60 points). The total runtime for each technology is about 30 minutes. The results are shown in Figure 9.11. For each technology, one can identify the four PFs relative to the values of $C_L$. The colored regions indicate the domain of validity for prediction.

The same statistical procedure is performed to validate the predictive models. We report in Figure 9.12 the histograms for the AMS 0.35µm technology. The prediction error for GBW is below 1.5%, with 50% of the designs deviating less than 0.25%. For the entire set of metrics, errors are well distributed around zero.

This experiment demonstrates that, even for a complex block, the inter-

Figure 9.10: Predictive model for the OTA
Increasing load

1.5pF

500fF

Figure 9.11: Bi-objective Pareto Fronts for the OTA circuit considering 4 values of load capacitance. The characterization is run for two technologies.

Figure 9.12: Histograms of the prediction error (%) for a total of 500 designs of the OTA circuit.
9.4 Non-monotonic Pareto Front

9.4.1 Description

The behavior of the predictive synthesis flow is tested for the case where the PF has dominated regions. In the proposed framework, this type of PF is modeled as a continuous function, covering both the Pareto-optimal and the dominated regions. The following experiment investigates the effect that a block presenting such a PF has on the optimization of a system.

For this experiment, we use a hierarchical systems based on analytical equations. This allows us to create an optimization problem that stresses the point we want to investigate. Three bi-objective problems are considered. Two problems represent low-level blocks, the third one the system (Figure 9.13). For convenience, parameters and performance metrics across the hierarchy are denoted with alphabet letters in this analysis. The three problems are formalized as the maximization or minimization of the performance metrics subject to constraints:

\[
\text{SYS : Sum and product of low-level performance}
\]

\[
\begin{align*}
\text{maximize: } & a_1 = b_1 \cdot c_1 \\
\text{minimize: } & a_2 = b_2 + c_2
\end{align*}
\]
POW: Monovariable, bi-objective function
maximize: $c_1 = 0.1 + e_1$
minimize: $c_2 = 0.1 + e_1^4(1-e_1)$
subject to: $e_1 \in [0, 1]$ (9.3)

TNK': A modified version of the TNK problem [KD01]
maximize: $b_1 = d_1$
minimize: $b_2 = d_2$
subject to: $(k - d_1)^2 + d_2^2 - 1 - 0.1 \cdot \cos(16 \cdot \arctan(\frac{k-d_1}{d_2})) \geq 0$ (9.4)
$(k - d_1 - 0.5)^2 + (d_2 - 0.5)^2 - 0.5 \leq 0$
$d_1 \in [0, \pi]; \quad d_2 \in [0, \pi]; \quad k = \frac{2\pi}{5}$

The TNK’ function has a non-monotonic PF due to the two constraint terms. POW and SYS, in turn, have a monotonic relation between their performance metrics. Next, we discuss the impact of the TNK’ function in the BU characterization and in the TD synthesis steps.

9.4.2 Results

The characterization of the low-level blocks is reported in Figure 9.14. The curves are discretized using 25 points and the NBI method is run using MATLAB’s SQP algorithm. The direction for improvement is indicated by the large arrows. In addition, we apply the change of variables proposed in this work (i.e. $f \rightarrow [u, t]$). The axis for the parameterization of the curves (u) is indicated by a dotted line. For the case of the TNK’ block (Figure 9.14.b), the disconnected Pareto-optimal regions are highlighted. Because the PF of TNK’ is non-monotonic, the predictive function $[b_1, b_2] = G_{TNK}(u_b)$ has oscillating local maxima and minima: A behavior that leads to difficulties in the subsequent characterization of the system.

In the experiments, the characterization process of SYS fails when the NBI method is executed using SQP and the jump-start technique. This happens because the SQP algorithm becomes trapped in the local minima of $G_{TNK}(u_b)$, which prevents it from moving from one optimal region to the
next one. To solve this situation, we use a hybrid global-local search algorithm. For each NBI problem, a single-objective GA is used first, followed by the SQP algorithm.

The results of this hybrid search are illustrated in Figure 9.15. The PS of the system (Figure 9.15b) shows a discontinuous evolution of the parameters $u_b, u_c$. Following the curves from left to right (i.e. from $u_a = -1$ to $u_a = 1$), the TNK’ parameter $u_b$ presents three major transitions. They correspond to abrupt jumps between two Pareto-optimal regions (Figure 9.14b). The parameter $u_c$ reacts to these transitions and also assumes a discontinuous behavior.

To verify the impact of these discontinuities on the quality of the prediction, the TD synthesis process is run for 100 points. The validated points follow...
Figure 9.16: Synthesis of 100 points. The validated PF of SYS (a) and the statistical error (b)

the predicted PF closely, except for three regions, where errors attain values around 20% (Figure 9.16). These localized but large errors appear in the transitions between optimal regions in the TNK’ block as highlighted in Figure 9.17. However, the figure illustrates that most of the points that contribute to the PF of SYS lie in the optimal regions of the TNK’.

In summary, the presence of a non-monotonic PF in the hierarchy of the system impacts the design flow in the following ways:

Figure 9.17: Synthesis of 100 points for the TNK’ block. The transition between optimal regions is responsible for the prediction errors at the system-level.
1. It introduces local minima/maxima in the optimization problems. Therefore, global search methods have to be used in the NBI method.

2. It introduces discontinuities in the PS of the system. This affects the sizing of all blocks in the hierarchy, as they must react to sudden changes to one parameter.

3. These discontinuities lead to large prediction errors localized at the transitions between Pareto-optimal regions.

Nevertheless, the experiment demonstrates that high-level predictive models generated under these circumstances are trustworthy. The predictive synthesis process presented a statistical error of less than 1% for more than 80% of the simulated design points. This accuracy is expected to further improve if we increase the number of discretization points around the discontinuities (i.e. make the transitions in Figure 9.15b to be steeper).
10 Conclusion

10.1 Summary

Mixed-domain/mixed-technology designs are characterized by the use of specialized description languages and simulation tools for each of the domains. To integrate these flows in the same design process, the adoption of a hierarchical approach with multiple levels of abstraction becomes pivotal. The use of high-level modeling languages allows to group the behavior of blocks from different domains in the same description. The process of design consists of traversing this hierarchy and making trade-off decisions at each level.

This chapter was dedicated to the development of a hierarchical design methodology allowing a systematic evaluation of performance trade-offs in the design of multi-physics systems. The framework we developed is based on the emerging idea of using optimal trade-off curves (Pareto Fronts) as a suitable abstraction for the behavior of a block. The process is based on the characterization of trade-off curves for each block and their bottom-up propagation to compute system-level trade-offs. The process of characterization is performed through numerical optimization and is computationally expensive. The contribution presented in this work was motivated by the opportunity to extend this technique to leverage the data collected during this characterization process. By the use of interpolation techniques on the design data, it becomes possible to perform the automatic synthesis of such heterogeneous systems: from high-level performance requirements down to a prediction of the sizing of every low-level block.

The design framework was presented in chapter 8. A first section presented the complete framework. The concept of a predictive model that creates a continuous map between performance requirements and sizing parameters was formalized. We then showed how this predictive models fit seamlessly in any optimization based design process and provide a self-contained and language agnostic representation of block, which enables portability and design reuse.
10.2. PERSPECTIVES

In the same chapter, two sections were devoted to the algorithms and mathematical methods needed to build the predictive models. Based on the constructive properties of the NBI method, we can derive a change of variables and tests that ensure the existence of functions interpolating the design data.

Finally, the hierarchical synthesis flow was tested on three test cases. Two of them being real-world applications and the third being a set of analytical equations used to stress the framework. For all test cases, we have performed a statistical analysis and showed that the automatic synthesis flow can provide reliable predictions of the sizing parameters.

10.2 Perspectives

Future work has to be done in the following areas to further extend the applicability of the method and its computational efficiency.

- Incremental improvement of the predictive models: in general, the number of samples needed to correctly represent a PF is not known in advance. Choosing an arbitrary value leads either to an over- or undersampling, and therefore to a waste in computational resources or a large prediction error, respectively. We can improve the characterization process to add points to the models in an incremental fashion. This would allow us to start with a coarse predictive model and, every time a large prediction error is detected, make a local improvements to the model by running the NBI method on the region where the prediction is poor.

- Further develop the notion of environment parameters: in the performed experiments, environment parameters concerned one single block. It is important to investigate how to manage environment parameters coupling two blocks in an assume/guarantee relationship. In particular we want to investigate how the approach proposed by Sun [Sun11] can be transposed in our predictive framework.

- Extend the exploration to digital (discrete) blocks: digital circuits are pivotal to build complex heterogeneous systems. Digital processing allows to control and extend the functionality of AMS blocks. To raise
the level of abstraction that can be handled in the framework, a connection with digital design is pivotal. To this point we have to develop efficient exploration tools that are based on mixed-integer optimization in order to deal with the continuous sizing of AMS blocks and the discrete choices of digital components. This is the most important and challenging topic in our future research on predictive synthesis via data mining methods.
Advanced packaging technologies such as 3D-IC enable the construction of highly complex systems by means other than device miniaturization. By splitting the functionality of a chip into multiple dies, the thight integration of heterogeneous components becomes possible. This opens the doors to novel and countless applications involving combinations of sensors, memories, analog, digital, and RF circuits. However, to take full advantage of these technologies and create high-performance low-power applications that meet time-to-market requirements, designers must be supported by efficient system-level exploration methods.

The handling of thermal variation and noise coupling; the organization of the interconnect structure; the development of new design and cross-domain partitioning methods for heterogeneous systems; the incorporation of physical information at the system-level; are all more relevant than ever. In the course of this thesis, we investigated two major concerns of 3D design: the partitioning of blocks into multiple layers in terms of area, temperature and communication constraints; and the development of design methods that fit the need for system-level exploration of heterogeneous contents.

The thesis was organized in two distinct and independent parts. Nevertheless, it is evident that both system-level partitioning and hierarchical design have to converge in order to offer a holistic design exploration environment (Figure 11.1). The contributions of this thesis on optimization strategies for block partitioning and systematic performance trade-off for hierarchical systems provide a set of automation tools that can enable this convergence.
Chapter 11. General Conclusion

Figure 11.1: Conceptual combination of partitioning and system synthesis
LIST OF PUBLICATIONS

International journals with review committee


Patents

- Felipe Frantz, Lioua Labrak, and Ian O’Connor. Procédé de conception optimisé de circuits électroniques intégrés, Filed November 2011

International conferences with review committee

- Felipe Frantz, Lioua Labrak, and Ian O’Connor. 3D-IC floorplanning: Applying meta-optimization to improve performance. VLSI and System-on-Chip (VLSI-SoC), IEEE/IFIP 19th International Conference, pp.404-409, October 2011
- V. Viswanathan, Lioua Labrak, Felipe Frantz, D. Navarro, and Ian O’Connor. Model based design of Imager using Abstraction- segre-
gated parameter dependency graphs. IEEE-NEWCAS Conference, June 2011. Bordeaux, France


**Posters sessions**

- Felipe Frantz, Lioua Labrak, and Ian O’Connor. A meta-optimization approach to setup a 3D thermal-aware floorplanner. DATE’11 Friday Workshop on 3D Integration, March 2011. Grenoble, France
- Felipe Frantz, Lioua Labrak, and Ian O’Connor. Methods and Tools for 3D Heterogeneous Design. DATE’10 Friday Workshop on 3D Integration, pp. 52-53, March 2010. Dresden, Germany


Global Semiconductor Alliance (GSA) 3DIC Working Group meeting, 2011.


[CC04] M.C. Campi and G. Calafiore. Multiple Participant Decision


[DA10] G. Druais, P. Ancey, L.L. Chapelon, J. Charbonnier, S. Chr- 


Thesis on Micro et Nanotechnologies 154 Felipe Frantz Ferreira / 2012 Ecole Centrale Lyon


[MOS] [http://www.mosis.com](http://www.mosis.com).


[SMBDM10] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli. Sun-


[UCL] [http://cadlab.cs.ucla.edu/three_d/3dic.html](http://cadlab.cs.ucla.edu/three_d/3dic.html).


[WID] JESD229 Wide I/O Single Data Rate (SDR).


[ZL07] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. Evolutionary


1.1. Miniaturization vs. diversification ........................................... 2
1.2. 3D Packaging technologies .................................................. 2
1.3. Example of a vertical stacking with 3 ICs ................................. 3
1.4. Formation of TSVs ............................................................... 3
1.5. Roadmap for TSV technology ................................................... 5
1.6. Market growth for sensors ....................................................... 6
2.1. Representation of a floorplan with constraint graphs .................... 14
3.1. General workflow of the floorplan tool ....................................... 18
3.2. Software structure ............................................................... 20
3.3. 3D move probability decreases with iteration number .................... 23
3.4. Workflow of the floorplan algorithm and the meta-optimization ....... 25
3.5. Area vs. wirelength Pareto front for one and two benchmarks ........... 26
3.6. The decision variable for the generic algorithm ............................ 27
4.1. Floorplan, constraint graph and sequence pair representation ......... 30
4.2. Area overhead of TSVs ......................................................... 31
4.3. Footprint measure is only sensitive to large dimensions .................. 32
4.4. W irelength minimization move and slack .................................. 33
4.5. The detailed model and its simplification ................................... 35
4.6. From physical structure to resistive models ................................ 36
4.7. Grid size in tile analysis ....................................................... 37
4.8. Grid sub-division into cells depends on grid dimensions and block sizes ................................................................. 37
4.9. Vertical heat flow and power distribution .................................... 39
4.10. Two phase algorithm, switching criteria, and the thermal profile ....... 39
4.11. NoC-based MPSoC ............................................................. 41
4.12. Features considered by MoNICA ............................................ 42
4.13. Translating a mapping to a floorplan ....................................... 43
4.14. Optimization flow with MoNICA ............................................ 44
4.15. The effect of a modification in the mapping on the optimal floorplan. 45
4.16. Blocks laid out in a grid and the equivalent SP diagram ................. 46
5.1. 2D floorplan solutions for the considered benchmarks .................. 49
5.2. Characteristics of the studied benchmarks ................................... 49
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3</td>
<td>Parameter set obtained via meta-optimization</td>
</tr>
<tr>
<td>5.4</td>
<td>Comparisons between mofp and Parquet</td>
</tr>
<tr>
<td>5.5</td>
<td>Graphical comparison based on the results from table 5.3</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison of runtime trends</td>
</tr>
<tr>
<td>5.7</td>
<td>The compromise between vias, area, and wirelength</td>
</tr>
<tr>
<td>5.8</td>
<td>Dimensions of the four layer stack</td>
</tr>
<tr>
<td>5.9</td>
<td>Initial configuration for NoC-aware floorplanning</td>
</tr>
<tr>
<td>5.10</td>
<td>Sample results of NoC-aware floorplanning</td>
</tr>
<tr>
<td>6.1</td>
<td>Overlay of hot spot routines and thermal properties of TSVs</td>
</tr>
<tr>
<td>7.1</td>
<td>V-cycle for system design</td>
</tr>
<tr>
<td>7.2</td>
<td>Heterogeneous design flow</td>
</tr>
<tr>
<td>7.3</td>
<td>A simulation model</td>
</tr>
<tr>
<td>7.4</td>
<td>Parameter space and performance space</td>
</tr>
<tr>
<td>7.5</td>
<td>Synopsis of an automatic design process</td>
</tr>
<tr>
<td>7.6</td>
<td>The parameter space $\mathcal{X}$ and the performance space $\mathcal{F}$</td>
</tr>
<tr>
<td>7.7</td>
<td>Individual blocks integrated in a system</td>
</tr>
<tr>
<td>7.8</td>
<td>Relation between hierarchical levels in the design process</td>
</tr>
<tr>
<td>7.9</td>
<td>The five steps in state of the art hierarchical design</td>
</tr>
<tr>
<td>7.10</td>
<td>Bottom-up combination of Pareto Fronts</td>
</tr>
<tr>
<td>7.11</td>
<td>Top-down design using Pareto Fronts</td>
</tr>
<tr>
<td>7.12</td>
<td>Obtaining a Pareto Front by combination or interpolation</td>
</tr>
<tr>
<td>7.13</td>
<td>Interpolation of the performance vector $\mathbf{f}$ to obtain $\mathbf{x}$</td>
</tr>
<tr>
<td>8.1</td>
<td>Global view of the synthesis framework</td>
</tr>
<tr>
<td>8.2</td>
<td>Characterization of Pareto-fronts</td>
</tr>
<tr>
<td>8.3</td>
<td>Models (simulation and predictive) established on optimal design data</td>
</tr>
<tr>
<td>8.4</td>
<td>Illustration of the prediction functions in two and three dimensions</td>
</tr>
<tr>
<td>8.5</td>
<td>Interface including the output value of the validity function</td>
</tr>
<tr>
<td>8.6</td>
<td>Scheme to optimize a system that is implemented by two blocks</td>
</tr>
<tr>
<td>8.7</td>
<td>Encapsulating systems of growing complexity</td>
</tr>
<tr>
<td>8.8</td>
<td>Overview of the flow for multiple hierarchical levels</td>
</tr>
<tr>
<td>8.9</td>
<td>Approaches to compute the Pareto Front</td>
</tr>
<tr>
<td>8.10</td>
<td>The NBI method in two and three dimensions</td>
</tr>
<tr>
<td>8.11</td>
<td>Definition of single-objective problems in BI methods</td>
</tr>
<tr>
<td>8.12</td>
<td>Visual representation of different formulations of the BI problem</td>
</tr>
<tr>
<td>8.13</td>
<td>Distance between NBI trajectories with normalized performance metrics</td>
</tr>
<tr>
<td>8.14</td>
<td>Strategies for the parallel resolution of NBI problems</td>
</tr>
<tr>
<td>8.15</td>
<td>Use of the multiobjective method results in a bijective map</td>
</tr>
<tr>
<td>8.16</td>
<td>Three functions constituting a predictive model</td>
</tr>
<tr>
<td>Figure Number</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>8.17</td>
<td>PF-shape limits choice of interpolation variables</td>
</tr>
<tr>
<td>8.18</td>
<td>Parameterization of changing variables</td>
</tr>
<tr>
<td>8.19</td>
<td>Projections of the Pareto Front on different planes</td>
</tr>
<tr>
<td>8.20</td>
<td>The mixed use of $G(F')$ and $G(u)$ in the synthesis flow</td>
</tr>
<tr>
<td>8.21</td>
<td>Simplicial forms used to discretize 2D, 3D and 4D Pareto Fronts</td>
</tr>
<tr>
<td>8.22</td>
<td>Comparing orientations for parameterization</td>
</tr>
<tr>
<td>8.23</td>
<td>Testing patches for a 3D Pareto Front</td>
</tr>
<tr>
<td>8.24</td>
<td>RBF interpolation example</td>
</tr>
<tr>
<td>8.25</td>
<td>Definition of a valid domain using linear constraints</td>
</tr>
<tr>
<td>8.26</td>
<td>Constructing the validity function using RBF interpolation</td>
</tr>
<tr>
<td>8.27</td>
<td>Characterization of a block for an environment parameter</td>
</tr>
<tr>
<td>9.1</td>
<td>Verification process used in the experiments</td>
</tr>
<tr>
<td>9.2</td>
<td>Optical link for Network-on-Chip application</td>
</tr>
<tr>
<td>9.3</td>
<td>Optical link predictive model and its hierarchy</td>
</tr>
<tr>
<td>9.4</td>
<td>Trade-off curves for each component</td>
</tr>
<tr>
<td>9.5</td>
<td>Contour plot for the Pareto Front of the optical link</td>
</tr>
<tr>
<td>9.6</td>
<td>Interpolation of optical link sizing Pareto Set</td>
</tr>
<tr>
<td>9.7</td>
<td>Domain of validity (volume) for the interpolation between two cases</td>
</tr>
<tr>
<td>9.8</td>
<td>Synthesis error for random designs inside the validity domain</td>
</tr>
<tr>
<td>9.9</td>
<td>Schematic of the OTA circuit</td>
</tr>
<tr>
<td>9.10</td>
<td>Predictive model for the OTA</td>
</tr>
<tr>
<td>9.11</td>
<td>Bi-objective Pareto Fronts for the OTA circuit</td>
</tr>
<tr>
<td>9.12</td>
<td>The prediction error of the OTA circuit</td>
</tr>
<tr>
<td>9.13</td>
<td>Hierarchy of the system</td>
</tr>
<tr>
<td>9.14</td>
<td>Block characterization</td>
</tr>
<tr>
<td>9.15</td>
<td>System characterization</td>
</tr>
<tr>
<td>9.16</td>
<td>Synthesized points for SYS</td>
</tr>
<tr>
<td>9.17</td>
<td>Synthesis of 100 points for the TNK' block</td>
</tr>
<tr>
<td>11.1</td>
<td>Conceptual combination of partitioning and system synthesis</td>
</tr>
</tbody>
</table>
List of Tables

3.1. Detailed list of variables for the meta-optimization ............ 28
5.1. Meta-optimization variables ........................................ 50
5.2. Performance comparison in 2D ..................................... 51
5.3. Footprint, wirelength and 3D-vias minimization for 4 layers stack . 52
5.4. Results considering TSV dimensions (averaged over 100 runs) ...... 55
5.5. Reference values from area-wirelength minimization ............... 58
5.6. Grid size for thermal simulation .................................... 58
5.7. Thermal-aware floorplanning results ................................ 58
5.8. NoC-aware floorplanning results .................................... 62

8.1. Comparison of parameterization choices ........................... 110
8.2. Typical functions for RBF interpolation with \( r = \|y - y_i\| \) ....... 115

9.1. Optical link trade-off problem ...................................... 126
9.2. Characterization of the components of the optical link .......... 127
9.3. Comparison between the construction of high-level predictive models and the MUBU approach. ......................... 128
9.4. Setup for the characterization of the OTA design trade-offs .... 131
ACRONYMS

API Application Programming Interface.

DSE Design Space Exploration.

EA Evolutionary Algorithms.

GA Genetic Algorithm.

IC Integrated Circuit.

MO Multiobjective.

MOEA Multiobjective Evolutionary Algorithm.

NBI Normal Boundary Intersection.

NoC Network-on-chip.

PF Pareto Front.

PS Pareto Set.

RBF Radial Basis Function.

SA Simulated Annealing.

SiP System-in-Package.

SO Single-objective.

SoC System-on-Chip.

SP Sequence Pair.

TSV Through Silicon Via.
Liste des personnes Habilitées à Diriger des Recherches en poste à l’Ecole Centrale de Lyon

<table>
<thead>
<tr>
<th>Nom-Prénom</th>
<th>Corps grade</th>
<th>Laboratoire ou à défaut département</th>
<th>Etablissement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEROUAL Abderrahmane</td>
<td>professeur</td>
<td>AMPERE</td>
<td>ECL</td>
</tr>
<tr>
<td>BURET François</td>
<td>professeur</td>
<td>AMPERE</td>
<td>ECL</td>
</tr>
<tr>
<td>JAFFREZIC-RENAULT Nicole</td>
<td>directeur de recherche</td>
<td>AMPERE</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>KRÄHENBUHL Laurent</td>
<td>directeur de recherche</td>
<td>AMPERE</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>NICOLAS Alain</td>
<td>professeur</td>
<td>AMPERE</td>
<td>ECL</td>
</tr>
<tr>
<td>NICOLAS Laurent</td>
<td>directeur de recherche</td>
<td>AMPERE</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>SCORLETTI Gérard</td>
<td>professeur</td>
<td>AMPERE</td>
<td>ECL</td>
</tr>
<tr>
<td>SIMONET Pascal</td>
<td>directeur de recherche</td>
<td>AMPERE</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>VOLLAIRE Christian</td>
<td>professeur</td>
<td>AMPERE</td>
<td>ECL</td>
</tr>
</tbody>
</table>

**Nbre Ampère 9**

| HELLOUIN Yves          | maître de conférences    | DER EEA                             | ECL          |

**Nbre DER EEA 1**

| GUIRALDENQ Pierre      | professeur émérite       | DER STMS                            | ECL          |
| VINCENT Léo            | professeur               | DER STMS                            | ECL          |

**Nbre DER STMS 2**

| LOHEAC Jean-Pierre     | maître de conférences    | ICJ                                 | ECL          |
| MAITRE Jean-François   | professeur émérite       | ICJ                                 | ECL          |
| MARION Martine         | professeur               | ICJ                                 | ECL          |
| MIRONESCU Elisabeth   | professeur               | ICJ                                 | ECL          |
| MOUSSAOUI Mohand      | professeur               | ICJ                                 | ECL          |
| MUSY François          | maître de conférences    | ICJ                                 | ECL          |
| ZINE Abdel-Malek      | maître de conférences    | ICJ                                 | ECL          |

**Nbre ICJ 7**

<p>| CALLARD Anne-Ségolène     | professeur               | INL                                 | ECL          |
| CLOAREC Jean-Pierre      | maître de conférences    | INL                                 | ECL          |
| GAFFIOT Frédéric         | professeur               | INL                                 | ECL          |
| GAGNAIRE Alain           | maître de conférences    | INL                                 | ECL          |
| GARRIGUES Michel         | directeur de recherche   | INL                                 | CNRS/ECL     |
| GENDRY Michel            | directeur de recherche   | INL                                 | CNRS/ECL     |
| GRENET Geneviève         | directeur de recherche   | INL                                 | CNRS/ECL     |
| HOLLINGER Guy            | directeur de recherche   | INL                                 | CNRS/ECL     |
| KRAWCZYK Stanislas       | directeur de recherche   | INL                                 | CNRS/ECL     |
| LETARTRE Xavier          | chargé de recherche      | INL                                 | CNRS/ECL     |
| O’CONNOR Ian             | professeur               | INL                                 | ECL          |
| PHANER-GOUTORBE Magali   | professeur               | INL                                 | ECL          |
| ROBACH Yves              | professeur               | INL                                 | ECL          |
| SAINT-GIrons Guillaume   | chargé de recherche      | INL                                 | CNRS/ECL     |
| SEASSAL Christian       | directeur de recherche   | INL                                 | CNRS/ECL     |</p>
<table>
<thead>
<tr>
<th>Nom</th>
<th>Titre</th>
<th>Institution 1</th>
<th>Institution 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOUTEYRAND Eliane</td>
<td>directeur de recherche</td>
<td>INL</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>TARDY Jacques</td>
<td>directeur de recherche</td>
<td>INL</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>VIKTOROVITCH Pierre</td>
<td>directeur de recherche</td>
<td>INL</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>CHEN Liming</td>
<td>professeur</td>
<td>LIRIS</td>
<td>ECL</td>
</tr>
<tr>
<td>DAVID Bertrand</td>
<td>professeur</td>
<td>LIRIS</td>
<td>ECL</td>
</tr>
<tr>
<td>BAILLY Christophe</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>BERTOGLIO Jean-Pierre</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>BLANC-BENON Philippe</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>BOGEY Christophe</td>
<td>chargé de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>CAMBON Claude</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>CARRIERE Philippe</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>CHAMPOUSSIN J-Claude</td>
<td>professeur émérite</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>COMTE-BELLOT geneviève</td>
<td>professeur émérite</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>FERRAND Pascal</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>GALLAND Marie-Annick</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>GODEFERD Fabien</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>GOROKHOVSKI Mikhail</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>HENRY Daniel</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>JEANDEL Denis</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>JUVE Daniel</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>LE RIBAULT Catherine</td>
<td>chargé de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>LEOBEOUF Francis</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>PERKINS Richard</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>ROGER Michel</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>SCOTT Julian</td>
<td>professeur</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>SHAO Liang</td>
<td>directeur de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>SIMOENS Serge</td>
<td>chargé de recherche</td>
<td>LMFA</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>TREBINJAC Isabelle</td>
<td>maître de conférences</td>
<td>LMFA</td>
<td>ECL</td>
</tr>
<tr>
<td>BENAYOUN Stéphane</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>CAMBOU Bernard</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>COQUILLET Bernard</td>
<td>maître de conférences</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>DANESCU Alexandre</td>
<td>maître de conférences</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>FOUVRY Siegfrid</td>
<td>chargé de recherche</td>
<td>LTDS</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>GEORGES Jean-Marie</td>
<td>professeur émérite</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>GUERRET Chrystelle</td>
<td>chargé de recherche</td>
<td>LTDS</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>HERTZ Dominique</td>
<td>past</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>ICHCHOU Mohamed</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>JEZEQUEL Louis</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>JUVE Denysse</td>
<td>ingénieur de recherche</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>KAPSA Philippe</td>
<td>directeur de recherche</td>
<td>LTDS</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>LE BOT Alain</td>
<td>directeur de recherche</td>
<td>LTDS</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>LOUBET Jean-Luc</td>
<td>directeur de recherche</td>
<td>LTDS</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>MARTIN Jean-Michel</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>MATHIA Thomas</td>
<td>directeur de recherche</td>
<td>LTDS</td>
<td>CNRS/ECL</td>
</tr>
<tr>
<td>MAZUYER Denis</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>PERRET-LIAUDET Joël</td>
<td>maître de conférences</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>SALVIA Michelle</td>
<td>maître de conférences</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>SIDOROFF François</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>SINOU Jean-Jacques</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>STREMSDOERFER Guy</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>Nom</td>
<td>Titre</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>THOUVEREZ Fabrice</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>TREHEUX Daniel</td>
<td>professeur</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
<tr>
<td>VINCENS Eric</td>
<td>maître de conférences</td>
<td>LTDS</td>
<td>ECL</td>
</tr>
</tbody>
</table>

**Nbre LTDS 25**

| Total Hdr ECL       | 91                         |
AUTORISATION DE SOUTENANCE

Vu les dispositions de l’arrêté du 7 août 2006,

Vu la demande du Directeur de Thèse

Monsieur I. O’CONNOR

et les rapports de

Monsieur H. GRAEB
Professeur associé hdr - TU München - Arcisstr. 21 - 80333 München - ALLEMAGNE

Et de

Monsieur F. PECHEUX
Maître de Conférences hdr - UPMC-LIP6 - 4 place Jussieu - 75252 PARIS

Monsieur FRANTZ FERREIRA Felipe

est autorisé à soutenir une thèse pour l'obtention du grade de DOCTEUR

Ecole doctorale ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE

Fait à Ecully, le 22 octobre 2012

P/Le directeur de l’E.C.L.
La directrice des Etudes

[Signature]

M-A GALLAND